



RWS INFORMATIE - ONGECLASSIFICEERD

## **TOPAAS: Deel 3 Modelonderbouwing (Informatief)**

|        |               |
|--------|---------------|
| Datum  | 28 maart 2018 |
| Status | Definitief    |



## Colofon

|                             |                                                                                                |
|-----------------------------|------------------------------------------------------------------------------------------------|
| Naam Standaard              | TOPAAS                                                                                         |
| Beschrijving:               | TOPAAS beschrijft een structurele aanpak voor faalkansanalyse van software intensive systemen. |
| Status:                     | Definitief                                                                                     |
| Datum                       | 28 maart 2018                                                                                  |
| Versienummer:               | 3.0                                                                                            |
| Soort:                      | Informatief                                                                                    |
| Verantwoordelijke PE:       | Jean-Luc Beguin                                                                                |
| Gebruik in proces:          | Aanleg en Onderhoud                                                                            |
| Netwerk:                    | HVWN, HWS en HWN                                                                               |
| Object:                     | Alle RWS-infrastructuur                                                                        |
| Hoofdkennisveld:            | Assetmanagement                                                                                |
| Kennisveld:                 | Risicogestuurd Beheer en Onderhoud (RGO)                                                       |
| Informatie:                 | Probo@rws.nl                                                                                   |
| Verantwoordelijke afdeling: | RWS GPO – Afdeling Advies Technisch Management (ATM)                                           |
| WW RWS Nummer:              | 1319                                                                                           |

## Overzicht wijzigingen

| Versie | Datum        | Wijzigingen                                                                                   |
|--------|--------------|-----------------------------------------------------------------------------------------------|
| 2.0    | 1 april 2011 | Definitief                                                                                    |
| 3.0    | 28 mrt 2018  | Tekstuele update, model is ongewijzigd, toelichting is uitgebreider, zie commentaaroverzicht. |



## Inhoud

|          |                                                    |
|----------|----------------------------------------------------|
| <b>1</b> | <b>Inleiding 7</b>                                 |
| 1.1      | Doel en scope 7                                    |
| 1.2      | De noodzaak tot modelmatige onderbouwing 7         |
| 1.3      | Structuur document 7                               |
| <b>2</b> | <b>Doelstellingen TOPAAS 8</b>                     |
| <b>3</b> | <b>Basisconcept TOPAAS 9</b>                       |
| <b>4</b> | <b>Software falen binnen foutenboom analyse 11</b> |
| 4.1      | Software falen en andere faalobjecten 11           |
| 4.2      | Beschouwde faalmodi 12                             |
| 4.3      | Definities binnen TOPAAS 14                        |
| 4.4      | Integratie in foutenboom 16                        |
| <b>5</b> | <b>Basisopzet parametermodel 18</b>                |
| 5.1      | Methodische opzet 18                               |
| 5.2      | Basisfaalkans 18                                   |
| 5.3      | Afronding van de resultaten 19                     |
| 5.4      | Boven en ondergrens van de methode 19              |
| 5.5      | Praktische invulling 19                            |
| <b>6</b> | <b>Onderbouwing vragenlijst 21</b>                 |
| 6.1      | Totstandkomingsproces 21                           |
| 6.2      | Software product 24                                |
| 6.3      | Requirements traceability/verifieerbaarheid 25     |
| 6.4      | Testen 26                                          |
| 6.5      | Executieomgeving/gebruik 26                        |
| <b>7</b> | <b>Uitkomsten onderhoudsslagen 28</b>              |
| 7.1      | Eerste onderhoudsslag (2008-2017) 28               |
| 7.2      | Aandachtspunten 30                                 |
| <b>8</b> | <b>Vergelijking met de referentiedatabase 32</b>   |
| 8.1      | Referentiedatabase 32                              |
| 8.2      | Globale analyse 32                                 |
| 8.3      | Gedetailleerde analyse 34                          |
| <b>9</b> | <b>Referenties 50</b>                              |



# 1 Inleiding

## 1.1 Doel en scope

TOPAAS bestaat uit de onderstaande documenten:

| Deel | Naam                                     | Type            |
|------|------------------------------------------|-----------------|
| 1    | Handleiding                              | Kader (bindend) |
| 2    | Vragenlijst                              | Informatief     |
| 3    | Modelonderbouwing (voorliggend document) | Informatief     |
| 4    | Referentie- en validatiedatabase         | Informatief     |
| 5    | Onderhoudsproces (in concept)            | Informatief     |
| 6    | Audit framework (in concept)             | Informatief     |

Om een TOPAAS-analyse te maken zijn deel 1 Handleiding en eventueel deel 2 Vragenlijst benodigd. Indien men meer wil weten over de achtergrond van TOPAAS, dan dient deel 3 Modelonderbouwing geraadpleegd te worden. Deel 4 Referentie en validatiedatabase is de achterliggende database met de referentieprojecten, die benodigd waren voor het opzetten van het TOPAAS-model. Dit deel is alleen voor Rijkswaterstaat bedoeld. In deel 5 Onderhoudsproces staat omschreven hoe de documenten van het kader TOPAAS beheerd en onderhouden moeten worden door Rijkswaterstaat. Dit deel is alleen voor Rijkswaterstaat bedoeld. In deel 6 is een audit-framework omschreven. Hierin staat geschreven hoe een auditor een audit op een TOPAAS-analyse dient uit te voeren. Dit deel is nodig voor degene die een audit wil uitvoeren.

## 1.2 De noodzaak tot modelmatige onderbouwing

TOPAAS is een meetinstrument voor het bepalen van een faalkans, wat ook tot doel heeft wetenschappelijke beschouwing te kunnen weerstaan. Dit document heeft tot doel wetenschappers en onderhouders inzage te geven in de beslissingen achter het TOPAAS model.

Dit document zal met elke revisie van het TOPAAS model bijgewerkt moeten worden.

## 1.3 Structuur document

Hoofdstuk 2 geeft inzage in de doelstellingen van TOPAAS. Hoofdstuk 3 beschrijft het basisconcept achter TOPAAS: wat de generieke ideeën zijn achter TOPAAS, en hoe op een hoog niveau het model vervolgens is opgezet. Hoofdstuk 4 behandelt de definities zoals TOPAAS deze definieert. Hoofdstuk 5 introduceert het concept parametermodel, wat in hoofdstuk 6 uitgewerkt wordt met een concrete invulling. In hoofdstuk 7 worden de uitkomsten van de onderhoudsslag benoemd. Hoofdstuk 8 gaat dieper in op de kalibratie van het model.

## 2 Doelstellingen TOPAAS

TOPAAS is ontwikkeld om afnemers van software een kwantitatieve inzage te geven in de betrouwbaarheid van software, om het zo te kunnen plaatsen in RAMS analyses. Het gaat hier dan om gedrag dat de RAMS-prestaties in gevaar brengt, gegeven een degelijk ontwerp en een professionele realisatie.

Concreet streeft TOPAAS de volgende doelen na:

- Praktisch bruikbaar, dat wil zeggen
  - Het moet altijd mogelijk zijn een bruikbaar betrouwbaarheidsgetal te krijgen binnen redelijke doorlooptijd en budget grenzen, liefst voordat het systeem wordt ingezet in productiesituatie;
  - Bruikbaar voor zowel functionaliteit gerealiseerd met pakketsoftware COTS (Commercial off-the-shelf) als via maatwerk. Dit betekent de methode overweg moet kunnen met situatie waar niet alle idealiter gewenste gegevens beschikbaar<sup>1</sup> zijn voor de beoordelaar;
  - Bruikbaar voor alle types software, zeker voor software in te zetten voor hoge beschikbaarheid of veiligheidskritische toepassingen;
- Pragmatisch, wat zich uit in een relatief korte en eenvoudig in te vullen vragenlijst;
  - Betrouwbaar genoeg voor het doel, te weten
  - Het resultaat moet reproduceerbaar zijn;
  - De methode moet zich richten op relevante faalmodi, dat wil zeggen faalmodi die de RAMS-prestatie van het totale systeem in gevaar brengen. Faalmodi die geen effect hebben op RAMS-prestatie moeten buiten beschouwing gelaten worden;
  - De methode moet een resultaat leveren dat betrouwbaar genoeg is voor gebruik in een foutenboomanalyse. In het algemeen zijn één significant cijfer met een goede orde van grootte schatting voldoende.
- Algemeen geaccepteerd, dat wil zeggen:
  - Het resultaat wordt onderschreven door alle direct betrokkenen als een redelijke/eerlijke schatting van de betrouwbaarheid;
  - De aanpak is algemeen geaccepteerd als een van de beste methoden voor betrouwbaarheidsanalyses, in het licht van alle bovenstaande criteria.

---

<sup>1</sup> Dit kan komen doordat de leverancier de gegevens niet wil geven uit commerciële overwegingen of omdat de (meest proces)gegevens niet meer te achterhalen zijn.

### 3 Basisconcept TOPAAS

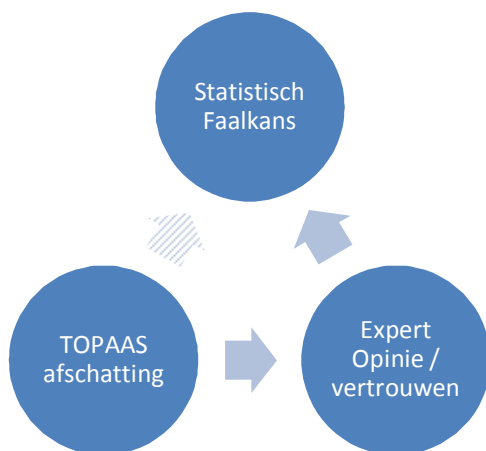
Statistische bepaling van de faalkans voor realistische systemen is in praktijk onhaalbaar, zeker als het om veiligheidskritische software gaat met een zeer kleine faalkans. Gegeven is dat een faalkans van software niet berekend kan worden op basis van bekende faalgegevens uit het veld of ervaring met de toepassing van de component op de specifieke installatie. Falen van software in het veld kan via problem management processen tot systematische verandering (met het doel van verbetering) leiden van het product, wat een andere faalkans tot gevolg heeft. Ook heeft de onderliggende complexiteit van de beslissingslogica het effect dat veelal het routinematige gedrag de ervaringsgegevens domineren, terwijl de uitzonderingssituaties de daadwerkelijke faalkans in praktijk bepalen. Het bepalen van een faalkans voor software op basis van interne en/of externe metrieken lijkt dan ook voor praktische veiligheidsgerelateerde systemen onhaalbaar [1].

TOPAAS doelt op het kwantificeren van het schijnbaar random manifesterende falen van software (zogenaamde Mandelbugs), binnen het gestelde inputdomein van de software. Hierbij wordt uitgegaan van de aanpak dat eenvoudig te detecteren deterministische fouten (zogenaamde Bohrbugs) door goed ontwerp en testen verwijderd zijn. Structurele configuratiefouten etc. vallen dus ook buiten TOPAAS.

Om tot een goede kwantificering te komen, hanteert TOPAAS dan ook geen statistisch concept van faalkans, maar wordt er een uitbreiding gemaakt naar de Bayesiaanse notie van faalkans. Deze Bayesiaanse notie accepteert dat de faalkans ook het vertrouwen van een expert in een systeem kan reflecteren. Dit is een aanpak die wordt gebruikt in meerdere praktische wetenschappelijke toepassingen, zie bijvoorbeeld [2, 3, 4]. Ook het Opschep-model [14] wat gebruikt wordt om de faalkans van menselijk handelen in foutenbomen te modelleren hanteert een vergelijkbare filosofie. De notie dat de faalkans van software wordt benaderd door het vertrouwen dat experts in het systeem hebben, is het centrale concept in de toepassing van TOPAAS. Door dit concept te accepteren, kan men in zeer onzekere omgevingen de betrouwbaarheid kwantificeren door het gebruiken van expert opinie [5]. Samenvattend, in een zeer onzekere omgeving waar statistische analyse onmogelijk is, wordt het vertrouwen van de expert als faalkans gebruikt.

Nu kent het gebruik van een enkele expert zijn beperkingen: er ontstaat subjectiviteit en het risico van uitval van dé expert. Om het minder subjectief te maken, wordt in het algemeen een groep van experts gebruikt [5, 6]. Deze aanpak is zeker niet uniek. Deze aanpak wordt vaker gebruikt om risico's te kwantificeren in omgevingen waar statistische data ontbreekt, bijvoorbeeld radiologie [7] en voedsel veiligheid [8]. Gezamenlijke Expert opinie is ook eerder toegepast als baseline voor de inschatting van software fouten [9], waarbij de aanpak iets conservatiever bleek ten opzichte van de uitkomsten met Reliability Growth Modelling.

Expert opinie is onder risico-analisten een geaccepteerde aanpak, maar er kleven praktische bezwaren aan: het is nogal omslachtig om telkens een groep bijeen te roepen, en er is een aanzienlijke kans dat experts het niet met elkaar eens worden over wat de faalkans uiteindelijk is.



Figuur 1 Relatie TOPAAS met de werkelijke faalkans

TOPAAS benadert de afschatting van een groep experts door middel van een parameter model. De factoren, zoals beschreven in Hoofdstuk 6 "*Onderbouwing Vragenlijst*", zijn dan ook de factoren die de experts zelf benoemd hebben als zijnde van invloed op softwarebetrouwbaarheid. De set van parameters is dus de geëxpliciteerde beoordelingsstructuur voor het vormen van een gezamenlijke mening van een specifieke set van experts, inclusief de weging in die mening. Dit staat enigszins los van de mening die andere experts kunnen hebben of zelfs wat de wetenschap van deze parameters vindt.

Er hebben op de lijst van factoren twee verificatieslagen plaats gevonden:

- Een kwalitatieve verificatie: in wetenschappelijke publicaties is gezocht naar de resultaten van vergelijkbare brainstormsessies door expertgroepen. In [10,11,12] zijn resultaten beschreven. Deze factoren zijn gedetailleerder beschreven (kennen meerdere factoren voor bij elkaar horende factoren met een redelijk sterke correlatie), maar de auteurs onderkennen geen categorieën die niet in het huidige TOPAAS-model onderkend worden.
- Een kwantitatieve verificatie: als controle zijn er 20 projecten door zowel de experts ingeschat, als aan de hand van de TOPAAS-scorelijst ingeschat. Hierdoor is dus bepaald hoe groot de afwijking is tussen de echte expert-opinie in de groep, en de resultaten van de TOPAAS-vragenlijst.

Hieruit blijkt dat er maximaal 1 orde grootte verschil tussen de experts en TOPAAS zit. Hiermee is het dus mogelijk om grove afschatting te maken van de faalkans van software ten behoeve van de RAMS-analyse.

## 4 Software falen binnen foutenboom analyse

### 4.1 Software falen en andere faalobjecten

De faalkans van software is een moeilijk grijpbaar begrip. In tegenstelling tot elektronische en mechanische componenten zijn er geen standaard modellen om sterkteberekeningen uit te kunnen voeren op basis van een software ontwerp (vergelijkbaar aan MIL-HDBK-217 voor elektronica).

In een foutenboom zal de detaillering van de faalmodi van software in lijn moeten zijn met het falen van hardwarecomponenten en menselijk handelen. Hiertoe wordt in de handleiding het begrip TUB (TaakUitvoeringsBlok) geïntroduceerd. Deze TUB is geïntroduceerd om niet vast te zitten aan een software indeling in modules of componenten en expliciet alleen de softwaredoorsnede mee te nemen die de taakuitvoering daadwerkelijk realiseert. Goede missie kritische software heeft namelijk altijd dubbelcheck/monitorende code in zich, die voor taakuitvoering straffeloos verwijderd zou kunnen worden. Bij de definitie van TUB speelden de volgende gedachten mee:

- Een TUB wordt gezien als de softwarematige analogie van een hardware component c.q. een menselijke taak.
- Het falen van een TUB lijkt in die zin op een hardwarematige component faalmode c.q. geen of onjuiste taakuitvoering.
- Net als bij hardware componenten en menselijke taken, moet de TUB een welgedefinieerde interactie hebben met andere hardware componenten, menselijke taken of TUBs.

Kijken we naar bekendere vormen van falen, namelijk hardware falen en menselijk falen, dan zijn er wel opvallende overeenkomsten en verschillen:

| Technisch (hardware) Falen                                                             | Menselijk Falen                                                                                             | Software Falen                                                                                                                                                              |
|----------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Falen is veelal random,<br><br>Wijziging van omgeving kan soms falen veroorzaken       | Falen kan random en systematisch zijn,<br><br>Wijzigen van omgeving kan falen veroorzaken                   | Falen is veelal systematisch <sup>2</sup> ,<br><br>Wijzigen van omgeving kan een belangrijke aanleiding tot falen zijn                                                      |
| Ontwerpfouten zijn mogelijk waar te nemen door de (versnelde) veroudering te monitoren | Ontwerpfouten, zoals in procedures en werkinstructies, afleesinstrumenten, etc. zijn mogelijk waar te nemen | Ontwerpfouten zijn moeilijk te detecteren als het laagfrequent (on demand) gedrag is: men kan niet door regelmatige inspectie zien of de component (binnenkort) gaat falen. |
| Verwacht gedrag buiten ontwerp specificaties is te voorspellen                         | Verwacht gedrag buiten ontwerp specificaties is enigszins te voorspellen                                    | Verwacht gedrag buiten ontwerp specificaties is lastig voorspelbaar                                                                                                         |
| Falen is voor een                                                                      | Falen is voor een                                                                                           | Softwarefalen is vrijwel                                                                                                                                                    |

<sup>2</sup> Zie §4.2 "Beschouwde faalmodi **Fout!** Verwijzingsbron niet gevonden." voor een nadere uitleg van de interpretatie hiervan.

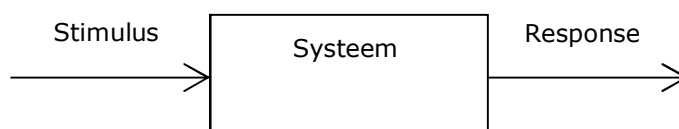
|                                                                             |                                                                                  |                                                                                                                            |
|-----------------------------------------------------------------------------|----------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| relatief groot deel merkbaar of merkbaar te maken door inspecties en testen | relatief groot deel onmerkbaar of lastig merkbaar te maken vanwege herstelgedrag | altijd onmerkbaar falen: de fout is altijd aanwezig, maar wordt vrijwel alleen gedurende aanspraak van een functie ontdekt |
| Door monitoren van gedrag kun je falen voorspellen                          | Door monitoren van gedrag kun je falen voorspellen                               | Falen van software kent geen voorspellende indicatoren                                                                     |
| Monitoring verkleint faalkans                                               | Monitoring/training verkleint faalkans                                           | Monitoring kan de faalkans vergroten (door toename complexiteit)                                                           |
| Kent geen herstelgedrag                                                     | Kent herstelgedrag                                                               | Kent redundantie en diagnose, maar slechts in zeer beperkte vorm herstelgedrag                                             |
| Bij identificatie van falen wordt component vervangen                       | Bij herkenning van falen wordt gedrag aangepast                                  | Bij identificatie falen wordt de component gerepareerd.                                                                    |

Uit bovenstaande vergelijking blijkt dat software meer overeenkomsten vertoont met menselijk falen dan met hardware falen. De voorgestelde aanpak voor de bijdrage van software aan het falen van een systeem zal dus logischerwijs erg lijken op die voor menselijk falen. Dus een focus op de taken en factoren equivalent aan trainingsniveau (kwaliteit van ontwerp en implementatie), uitvoeringsattitude (mate van interne monitoring en bewezen track record) en wijzigingen in de omgeving lijkt met een benadering die lijkt op menselijk falen (kijken naar het existentiële falen van handelingen in plaats van zoeken naar kwalitatieve redenen van falen) dan ook sterk op zijn plaats.

## 4.2 Beschouwde faalmodi

Software kan op verschillende manieren falen, maar er zijn keuzes te maken hoe deze faalmodi door TOPAAS beschouwd worden. In de TOPAAS aanpak beschouwen we falen als een gevolg van de oorzaken die in verschillende taxonomieën genoemd zijn, maar we bekommeren ons niet om de nog dieper liggende oorzaken hiervan. TOPAAS is dus primair gericht op het onderzoeken wat de kans van falen is, en niet wat de diepere onderliggende faalkans is. Hierdoor beschouwen we een TUB als een stimulus-response-systeem waarbij falen voort kan komen uit onjuiste inputs of een onjuiste verwerking van correcte inputs. De niet goed te observeren dieper liggende oorzaken van deze foutieve verwerking doen er dan niet meer toe. Dit is analoog aan veel modellen die gehanteerd worden voor afschatting van menselijk falen.

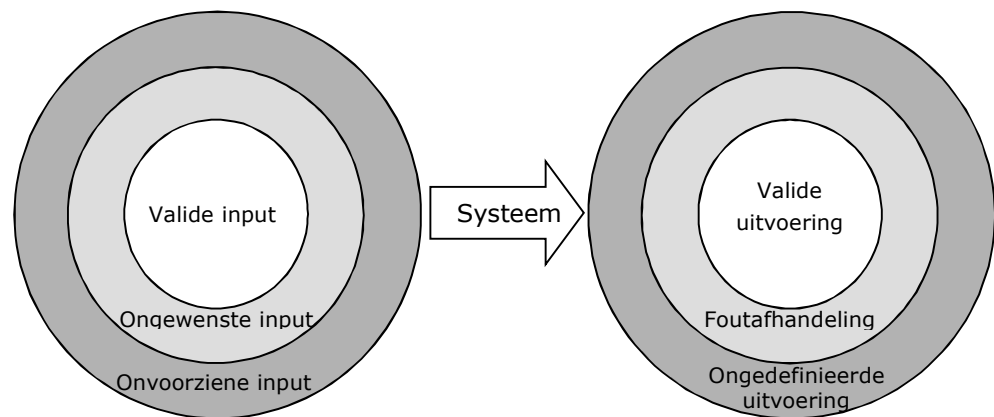
Als achtergrond bij het doorgronden van faal-modi, bekijken we het transformatie-probleem van input naar gewenste taakuitvoering (het waarnemen van signalen, het analyseren van die informatie, het vervolgens nemen van een beslissing en het geven van specifieke signalen). Een concept dat men ook ziet in de studie naar menselijk falen, waar men de mens modelleert als een organisch stimulus-response-systeem:



Als men de totale verzamelingen van inputs beschouwt, is het dus de taak van de specifieke TUB deze te transformeren naar taakuitvoering, waarbij de volgende zaken cruciaal zijn:

- valide invoer, welke bij correcte implementatie afgebeeld wordt op valide besturingsgedrag (correcte respons binnen de gestelde tijd), hetgeen tijdens testen van de TUB component dient te worden geverifieerd.
- voorziene ongewenste invoer, welke als het goed is, afgebeeld wordt op foutafhandeling en/of gewenst faalgedrag. Iets dat bij missiekritieke systemen ook expliciet wordt geverifieerd tijdens testen.
- Onvoorziene vormen van input, welke zullen leiden tot ongedefinieerd gedrag van het systeem.

Systematisch levert dit het volgende overzicht, waarbij het systeem wordt gemodelleerd als transformator van verzameling inputs op outputs:



Er wordt hier expliciet opgemerkt dat specifieke ontwikkelstrategieën verschuivingen in de omvang van deze verzamelingen teweeg kunnen brengen: door betere omgevingsanalyses of het gebruik van defensief programmeren [Dijkstra 1976] kan men de fractie van "onvoorziene input" verkleinen door de fractie "ongewenste input" te vergroten: het systeem kent een grotere ontwerptolerantie ten aanzien van ongewenste input waardoor deze robuuster wordt tegen foutieve data uit zijn omgeving. Consensus in de expertgroep is dat aan een defensieve strategie wel nadelen kleven: als men defensief programmeren te ver doorzet dan kan de applicatie door zijn toegenomen omvang en complexiteit als geheel wel eens onbetrouwbaarder worden.

Het gecombineerde domein van "valide input" en "voorziene ongewenste input" vormt het geaccepteerde inputbereik voor een TUB vanuit zijn specificatie: dit is zijn ontwerptolerantie. Dit bereik is normalerwijs onderdeel van de specificaties. Het domein van "onvoorziene vormen van input" is per definitie geen onderdeel van de specificaties (die is het complement daarvan t.o.v. het complete inputdomein).

Bij de faalkansmodellering van een TUB dient onderscheid gemaakt te worden in twee faalvormen:

- Falen ten gevolge van onvoorziene invoer. Dit is eigenlijk het gebruik van de TUB buiten zijn ontwerptoleranties, wat per definitie ongedefinieerde taakuitvoering van de TUB tot gevolg heeft. Een robuustheidsstrategie is hier primair bepalend of dit ook falen van de kritieke functies tot gevolg heeft. Met expliciete input check of een 'fail-to-safe mode' is er altijd

gedefinieerd gedrag van de TUB ook buiten zijn ontwerptoleranties. Het gebruik van een component buiten zijn specificaties kan komen door een onjuist/onvolledig vaststellen van de requirements en/of het onjuist/onvolledig uitvoeren van een omgevingsanalyse: het kan voorkomen dat verstoringen in omgevingssystemen tot onverwachte invoer leiden. Opgemerkt wordt dat in praktijk het lastig is om de omgeving volledig te modelleren bij een TUB met veel inputs: de analyse is in dat geval dus altijd minder volledig.

- Falen ten gevolge van een fout in de beslissingslogica (het intrinsieke falen van de component). Het gaat hier om alle foute handelingen die gevolg zijn van verwachte (valide en invalide) invoer die óf het verkeerde gedrag tot gevolg heeft óf het juiste gedrag op het verkeerde moment oplevert. Het gaat hier dus om fouten binnen de ontwerptoleranties van de TUB, waarbij een transformatie van de invoer zich dus binnen het domein van de valide/invalide invoer bevindt, maar wat niet resulteert in het gewenste gedrag om de kritieke functies uit te voeren. Zoals al is gemeld in de definitie kan het hier zowel gaan om het te lang uitblijven (of te vroeg komen) van gedrag of om het vertonen van het verkeerde gedrag.

TOPAAS richt zich expliciet op het intrinsieke falen van een component waarbij valide input tot te late/te vroege of ongewenste output leidt.

#### 4.3 Definities binnen TOPAAS

De gebruikte definities zijn overgenomen uit [13]. Centraal staat het begrip 'falen'. In de internationale standaarden zijn termen als failure, error en fault niet eenduidig gedefinieerd, zie bijvoorbeeld IEC 61508-4, §3.6 over de verschillende interpretaties. In TOPAAS wordt de volgende definitie gehanteerd:

**Falen** is een gebeurtenis, of een verzameling gebeurtenissen, waardoor een systeem zijn functionaliteit of een deel van zijn functionaliteit verliest.

Deze definitie van falen is expliciet gericht op het manifesteren van extern gedrag (in software wereld een vorm van 'fault' genoemd). Het Nederlandse begrip fout (of softwarefout) is in internationale (Engelse) terminologie verdeeld in 'error' (vergissing door een mens tijdens realisatie), 'defect' (afwijking van specificaties) en 'fault' (manifesterende gedragsafwijking).

Voor foutenboom analyse worden niet alle functies van een systeem in ogenschouw genomen. Bij foutenboomanalyse worden topevents ook wel aan een RAMS prestatie van een hoofdfunctie gekoppeld

Voorbeelden van hoofdfuncties zijn:

- De veilige sluiting van een waterkering bij een stormconditie;
- Het veilig routeren van een trein over een specifiek stuk spoor;
- Het veilig routeren van een vliegtuig;
- Het veilig besturen van een vliegtuig;
- Het veilig besturen van een schip.

Een **ongewenste topgebeurtenis**: (partieel) functieverlies van een systeem. De ongewenste topgebeurtenis (OTG) is de gebeurtenis, waarvan de kans wordt berekend in een kwantitatieve risicoanalyse. Veelal is dit het falen van een hoofdfunctie van het systeem.

- Voorbeelden van ongewenste topgebeurtenissen zijn: Niet sluiten van een waterkering bij aanwezige sluitcondities;
- Het ontstaan van een bijna-botsingssituatie op het spoor;
- Het ontstaan van een bijna-botsingssituatie in het vliegverkeer;
- Het verliezen van de controle over een vliegtuig gedurende de vlucht;
- Het verliezen van de besturing en/of aandrijving van een schip gedurende de vaart.

Uiteindelijke doel van de systemen is het leveren van de functionaliteit binnen de vereiste RAMS-prestatie. Deze kritieke functionaliteit die hiervoor nodig is, is als volgt gedefinieerd:

De **kritieke Functie** van een complex (software intensief) systeem is de uiteindelijke handeling die gerealiseerd moet worden door de samenwerkende systemen evt. bedienaars en de aangestuurde hardware.

De definitie van de systeemfuncties is het begin van elke foutenboom. Per functie worden de RAMS-prestaties geformuleerd in RAMS-eisen. Systeemfuncties, RAMS-prestaties en RAMS-eisen worden meestal ook alternatief geformuleerd in missies, topgebeurtenissen en missieduren

Een kritieke functie bestaat uit één of meerdere taakuitvoeringen van software: activiteiten die waarneembaar zijn en waarvan vastgesteld kan worden of deze goed of fout zijn uitgevoerd ten opzichte van het bereiken van bepaalde hogere orde doelen. In praktijk zijn dit zaken die door software worden geïnitieerd of bestuurd. Het (incorrect) uitvoeren van taakuitvoering is in principe ook een basisgebeurtenis in de foutenboom. Voorbeelden van taakuitvoeringen zijn:

- Sluitcommando geven van een kering;
- Sluiten van een specifieke klep;
- Noodventilatie starten in een tunnelbuis.

Een taakuitvoering kan worden gerealiseerd door één of meerdere TUBs. Een TUB is een technisch object en wordt gedefinieerd op basis van een gebalanceerd abstractieniveau door de taken die de software moet uitvoeren als basis te nemen. Een TUB wordt aldus als volgt gedefinieerd:

Een **TUB** is een (deel van een) taakuitvoering die door een afgebakende verzameling logische regels programmacode (of grafisch equivalent) wordt gerealiseerd, waarbij geldt dat:

- Een duidelijke afbakening te onderkennen is ten opzichte van andere stukken code én er een onderkende functionele doelstelling in het totaal van het systeem aanwezig is;
- De kwaliteitseigenschappen en taakuitvoering los te verifiëren zijn.

De verzameling logische regels wordt in zeer brede zin beschouwd:

- een TUB kan een specifieke applicatie zijn
- een TUB kan alle software op een dedicated hardware component (PLC) zijn,
- een TUB kan een proces of executable op een mainframe zijn
- een TUB kan een collectie samenwerkende processen in een systeem zijn;
- een TUB kan een deel van een applicatie/executable zijn die zich met een specifieke taakuitvoering bezig houdt.

Kern is dat een TUB een (deel van een) taakuitvoering realiseert, onderscheidbaar is en er enige mate van onafhankelijkheid is van de overige software.

Het falen van een TUB is als volgt gedefinieerd:

**Het falen van een TUB** is de (te lange) afwezigheid van gewenst gedrag, of het uitvoeren van verkeerd gedrag, door een TUB dat een bijdrage heeft aan de **ongewenste topgebeurtenis**.

Hiermee is het falen van de taakuitvoeringen van een TUB dus een Basic Event in de foutenboom van het systeem geworden.

Expliciet valt hieronder:

- Geen gedrag vertonen;
- Het verkeerde gedrag vertonen;
- Gedrag niet op het juiste tijdstip vertonen.

Nu het falen van een TUB is benoemd, kan men ook de faalkans van een TUB benoemen:

De **faalkans van een TUB** is waarschijnlijkheid dat taakuitvoering TUB faalt op basis van een vraag tot een gewenste taakuitvoering

Hier wordt expliciet benadrukt dat de faalkans wordt bepaald op basis van "een vraag tot gewenste taakuitvoering", hetgeen impliceert dat bij het uitblijven van de vraag het falen toegerekend moet worden aan het falen van de vraag-genererende component en als de vraag wel gesteld wordt eigenlijk per definitie al het daaropvolgende falen gerekend kan worden tot de bestudeerde TUB.

#### 4.4 Integratie in foutenboom

Eén van de kernproblemen met faalkansmodellering in foutenbomen is de vraag wat de (basis)gebeurtenissen zijn. Ook in de fysieke wereld is dit een vraagstuk: men kan modelleren dat een pomp niet werkt, maar men kan ook specifieker zijn door te modelleren dat één van de deelcomponenten van een pomp niet werkt.

Een basisgebeurtenis in een foutenboom behelst grofweg drie typen:

- falen van een hardware component;
- het niet, niet op het juiste moment of onjuist uitvoeren van een taak door de mens en;
- het niet, niet op het juiste moment of onjuist uitvoeren van een taak door software.

Zoals al eerder aangegeven zijn de benaderingswijzen voor de modellering van het detailniveau, vrijwel identiek voor al deze drie typen falen. De modelleeraanpak moet de balans in de boom waarborgendoor functies van subsystemen te benoemen, die via een functiedecompositie dan worden verdeeld in functies (taakuitvoeringen) van bovenstaande soort.

In de foutenboom dient ook functioneel herstel gemodelleerd te worden. Hierbij is "reparatie" van de softwarecomponent door middel van het uitvoeren van een handmatige werkinstructie een valide optie, en is het niet noodzakelijk de software zelf te repareren door middel van een bugfix. Voor software geldt net als bij

hardware dat er op gecontroleerde/procedurele wijze een reset kan worden gegeven of een component die de TUB implementeert wordt vervangen. Alhoewel een bugfix voor de uiteindelijke RAMS-prestaties vaak wel wenselijk is, is deze voor de korte-termijn functieherstel vaak dus niet noodzakelijk. TOPAAS geeft geen inschatting voor de betrouwbaarheidsbijstelling wegens het herstel van specifieke fouten in software.

## 5 Basisopzet parametermodel

### 5.1 Methodische opzet

De methodische basis van het TOPAAS model is een parametermodel, gelijkend op het OPSCHep-model voor menselijk falen. Het parametermodel gaat uit van een basiskans die voor alle software gelijk is.

Op basis van beschikbare informatie over de TUB wordt de basiskans bijgesteld. Hierbij wordt expliciet de nadruk gelegd op het beschikbaar zijn van informatie: informatie die niet beschikbaar is hoeft niet tot scoring te leiden omdat het TOPAAS model van nature voldoende conservatief is ingesteld zodat op basis van daadwerkelijk beschikbare informatie alleen tot scoring overgegaan hoeft te worden. Als men iets niet weet, kan men dus gewoon de scoring van dat onderwerp overslaan.

Voor elk stuk relevante informatie is een schaal bepaald waarop de TUB ingeschaald moet worden, resulterend in een vermenigvuldigingsfactor. De vermenigvuldigingsfactor kan groter of kleiner dan één zijn. Deze vermenigvuldigingsfactor is bepaald op basis van een veronderstelde correlatie met de faalkans. Praktisch wordt er gewerkt met de  $^{10}\log$  van deze factoren zodat die optellen tot de TOPAAS-score, welke via een tabel op basis van de aanspreekfrequentie omgezet kan worden in een faalkans per vraag of een beschikbaarheid.

Dus de basisfaalkans, vermenigvuldigd met een aantal verschillende factoren, bepaalt daarmee de inschatting van de faalkans van een TUB. De aanspreekfrequentie wordt meegenomen door de basisfaalkans afhankelijk te maken van deze frequentie ( $f$ ). Een simpele vorm van een parameter model zou zijn

$$P = P_B^f * F_1 * F_2 * \dots * F_n$$

Ofwel de kans ( $P$ ) is bepaald uit een basis faalkans  $P_B^f$  vermenigvuldigd met een reeks factoren.

In het TOPAAS model wordt de basisfaalkans ( $P_B^{1x/jaar}$ ) op 1 gezet (zie §5.2 "Basisfaalkans"). Alle  $F_i$ 's zijn dan vermenigvuldigingsfactoren die een veronderstelde invloed hebben op de faalkans.

### 5.2 Basisfaalkans

De faalkans per aanspraak is afhankelijk gemaakt van de aanspreekfrequentie. Als uitgangspunt wordt bij een aanspreekfrequentie van 1x/jaar de basisfaalkans 1 gekozen, omdat dit de meest conservatieve waarde is die men kan kiezen. Bij vergelijkbare modellen, bijvoorbeeld het OPSCHep model, wordt gekozen voor een gemiddelde faalkans. Zo'n gemiddelde is bij software echter onbepaald, daarom is voor een zeer conservatieve benadering gekozen. De keuze voor een basisfaalkans van 1 komt de onderhoudbaarheid van de methode ten goede, als op een later moment blijkt dat extra factoren toegevoegd moeten worden, zal de herijking van bestaande analyses minder vaak nodig blijken.

Gevolg is dat als men absoluut niets weet van de TUB, deze met een faalkans 1 in de foutenboom wordt opgenomen. Gezien de totale afwezigheid van enige

informatie en het niet uitgevoerd hebben van langdurige testen/simulaties (een parameter die een afnemer zelf kan beïnvloeden) lijkt dit geen onterechte aanname.

### 5.3 Afronding van de resultaten

Het natuurlijke gedrag van de experts is afronden naar de conservatieve kant. Een expert spreekt dan ook nooit van een schatting van  $10^{-2,60372}$ , maar van een geschatte faalkans van  $10^{-2}$ . Dit reflecteert enerzijds de conservatieve natuur van experts, maar ook de onmogelijkheid om überhaupt beter te schatten dan een hele ordegrootte. In praktijk is elke nauwkeuriger schatting dan een hele ordegrootte schijnnaauwkeurigheid.

Ook methoden als de IEC 61508 hanteren bandbreedtes van faalkansen om te benadrukken dat het bepalen van de betrouwbaarheid van software (nog) geen wetenschap is die vergelijkbaar is aan sterkteberekeningen van constructies.

TOPAAS imiteert dit natuurlijke gedrag van experts. Uit experimenten op de referentie database blijkt ook dat de fit tussen expertgroep en de TOPAAS uitkomsten significant slechter wordt als de conservatieve afronding niet wordt gehanteerd.

In praktijk blijkt dat dit conservatief afronden geen probleem vormt voor foutenbomen.

### 5.4 Boven en ondergrens van de methode

Deze ordegroottes zijn aan de onderkant begrensd door wat de industrie op dit moment als best haalbaar beschouwd: een betrouwbaarheid van  $10^{-5}$  op afroep. Dit is gebaseerd op industrie-concensus (IEC 61508, inleiding deel 1) enerzijds, en anderzijds ook de verhouding tot hardwarefalen (die meestal in de  $10^{-5}$  tot  $10^{-6}$  ranges is). De gedachte is dat de implementatie van een functie in software nooit betrouwbaarder zal functioneren dan de implementatie van die functie in een conventioneel relais. Ook de menselijke betrouwbaarheidsanalyse kent een expliciete grens van  $10^{-5}$ , welke uiteindelijk dominant is ten opzichte van softwarefalen. Software wordt door mensen ontworpen en gebouwd op basis van vooraf benodigde inzichten.

De bovengrens is uiteraard  $10^0$ . Er is een antwoorden set mogelijk die boven deze score uitkomt. Een TOPAAS score van 0 (faalkans 1) weerspiegelt het totale gebrek aan vertrouwen in de taakuitvoering en dient als RAMS bevinding te worden behandeld in de vorm van herstructureren van ontwerp en foutenboom. Omdat de faalkans bedoeld is voor gebruik in een foutenboom (en niet een gebeurtenissenboom) levert een TOPAAS score van -1 of hoger geen kwantificatie op. Achterliggende reden is dat de mathematische achtergrond van foutenbomen niet kan omgaan met relatief grote kansen.

Valide ordegroottes bij aanspreekfrequentie 1x/jaar zijn dus:  $10^{-2}$ ,  $10^{-3}$ ,  $10^{-4}$  en  $10^{-5}$ .

### 5.5 Praktische invulling

De generieke formule voor een parametermodel is de volgende

$$P = P_B^f * F_1 * F_2 * \dots * F_n$$

Alhoewel dit de normale benaderingswijze van parametermodellen is, wordt in praktijk in TOPAAS alleen de ordegraad van de faalkans van software afgeschat in een logaritmische schaal. Ook zijn, omwille van het gebrek aan nauwkeurigheid, alle beïnvloedingsfactoren ook in ordegraden gedefinieerd. Hierdoor is van  $P$  eigenlijk alleen de  $^{10}\log(P)$  van belang: dit is de exponent die de ordegraad van de faalkans aangeeft en gedetailleerdere informatie is niet noodzakelijk om tot een afschatting te komen.

Eigenlijk zijn we dus alleen maar geïnteresseerd in de exponent van de faalkans. Schrijven we de ordegraad van  $P$  als  $n$  (dus  $n = ^{10}\log P$ ) dan wordt de factorisering een redelijk eenvoudige optelling:

$$n = ^{10}\log P_B^f + \sum_i ^{10}\log F_i$$

Aangezien er alleen afschattingen worden gemaakt op basis van ordegraden, en de eenvoud van optellen boven een vermenigvuldiging, is er dan ook voor gekozen om de optelformule te hanteren voor de afschatting van de faalkans. Dit betekent dat de gehanteerde ordegraad van de uiteindelijk afgeschatte faalkans de optelling is van de  $^{10}\log$  van alle van toepassing zijnde factoren.

In het TOPAAS-model wordt dus uitgegaan van een optelling van de ( $^{10}\log$ ) exponent van de beïnvloedingsfactoren  $F_i$ . De  $^{10}\log$  van  $F_i$  noemen we  $B_i$ . Dit impliceert dus dat uiteindelijk de ordegraad van de faalkans is (uitgaande van een basisfaalkans  $P_B$  van 1, zoals beschreven in §5.2 "Basisfaalkans"):

$$P = 10^n \text{ met } n = \sum_i B_i$$

In de specifieke invulling van het parametermodel (zie Hoofdstuk 6 "Onderbouwing Vragenlijst") wordt er verder gebruik gemaakt van  $B_i$ -waarden.

## 6 Onderbouwing vragenlijst

Voor elke factor moeten de schalen  $B_i$  bepaald worden. Deze invulling van de schalen is gebaseerd op expliciete uitspraken van de experts. De experts zelf hebben dus een weging aangegeven in hun oordeel van het systeem, en hebben daar onderling consensus over bereikt.

### *Uitgangspunten*

Kerngedachte bij deze inschatting van factoren is dat de common practice in de industrie/toepassing het neutrale element oplevert (vermenigvuldigingsfactor 1, optelfactor 0). Dit is wederom een conservatieve inschatting: een partij die software ontwikkelt en zich dus conformeert aan de common practice (niet zijnde de best practice in de industrie), zal dus eindigen met een faalkans van 1. Dit symboliseert dat voor missie-/veiligheids-kritische systemen men beduidend meer effort moet doen dan de common practice in de IT-industrie te hanteren. Ook symboliseert dit dat men "gestraft" wordt door dingen te doen die tegen de common practice van de industrie ingaan. Dit heeft ook als bijkomend voordeel dat, als men geen inzage heeft in het proces, de praktische scoring (geen toevoeging waardering) overeenkomt met de common practice van de industrie.

De genoemde factoren hebben alle een redelijk eenvoudige schaal, waarbij elke factor nauwelijks rekening houdt met andere factoren. Binnen de huidige aanpak wordt alleen een beperkte afhankelijkheid tussen het SIL niveau en een aantal andere factoren gesteld. Dit is een bewuste aanpak om zo het model eenvoudig te houden.

Het kwalitatief afschatten van de faalkans van de TUB gebeurt via een score op de volgende factoren:

- Totstandkomingproces
- Product
- Requirements traceability/verifieerbaarheid
- Testen
- Executieomgeving/gebruik

De auteurs realiseren zich dat er sprake zou moeten zijn van een verminderde meeropbrengst als alle factoren zeer positief zijn: in een IEC61508 SIL-4 omgeving is bijvoorbeeld de extra toegevoegde waarde van zeer uitgebreide testperioden vrijwel verwaarloosbaar. Toch is er besloten alleen maatregelen uit te sluiten op SIL3/SIL4 niveau als de maatregel op dat moment expliciet wordt afgedwongen in de IEC61508. Doordat alle waarden redelijk conservatief worden ingeschat en er uiteindelijk alleen gekeken wordt naar de orde grootte van de faalkans, lijkt het de auteurs een goed offer ten gunste van de eenvoud.

### 6.1 Totstandkomingsproces

#### *Ontwikkelproces*

Het gebruik van de IEC 61508 heeft veel invloed op de uiteindelijke kwaliteit van het eindproduct.

| <b>1 Het ontwikkelproces voldoet aan één van de SIL's van de IEC 61508</b> |                                                                            |   |
|----------------------------------------------------------------------------|----------------------------------------------------------------------------|---|
| 1                                                                          | Onbekend, het ontwikkelproces voldoet niet aantoonbaar aan een SIL-niveau. | 0 |

|   |                                                                                                               |      |
|---|---------------------------------------------------------------------------------------------------------------|------|
| 2 | Ontwikkelp proces voldoet aantoonbaar niet aan een SIL-niveau door het gebruik van Not Recommended practices. | 1/2  |
| 3 | Ontwikkelp proces voldoet aantoonbaar aan SIL-1 niveau.                                                       | -1/2 |
| 4 | Ontwikkelp proces voldoet aantoonbaar aan SIL-2 niveau.                                                       | -1   |
| 5 | Ontwikkelp proces voldoet aantoonbaar aan SIL-3 niveau.                                                       | -2   |
| 6 | Ontwikkelp proces voldoet aantoonbaar aan SIL-4 niveau.                                                       | -3   |

Voor de schalen van het SIL-level is de invloed één orde grootte conservatiever ingeschat als dat de wereldwijde industrie-consensus aangeeft. Hier wordt bewust conservatiever ingeschat dan dat de IEC61508 zelf suggereert: een SIL-4 systeem zal op basis van deze meting een faalkans van  $10^{-3}$  krijgen. Dit is omdat het causale verband tussen het SIL-level en de betrouwbaarheid van het systeem niet wetenschappelijk is aangetoond, en het volgen van een kookboek geen goed resultaat garandeert.

Een minimale eis is wel dat er een basaal kwaliteitssysteem aanwezig is dat geschikt is voor softwareontwikkeling. De consensus binnen de expertgroep is dat de afwezigheid van een basaal kwaliteitssysteem, dat quality-gates hanteert, configuratiemanagement vereist en minimale vastlegging daarvan borgt grenst aan de minimale vereisten voor verantwoordelijke softwareontwikkeling. Afwezigheid daarvan impliceert dat men niet kan spreken van een goed ontworpen en getest systeem, waardoor TOPAAS niet toepasbaar is.

#### *Gebruik van inspecties*

Inspecties op documenten en code hebben een zeer sterke invloed op de kwaliteit van software. In praktijk zijn structurele reviews vaak (kosten)effectiever dan testen. Omdat in de IEC 61508 deze op SIL3 en SIL4 niveau al verplicht zijn, en Fagan inspecties aangeraden worden, worden deze op die SIL-niveaus anders gescoord.

| <b>2 Gebruik van inspecties</b> |                                                                                   | <b>Normaal</b> | <b>SIL3/SIL4</b> |
|---------------------------------|-----------------------------------------------------------------------------------|----------------|------------------|
| 1                               | Onbekend                                                                          | 0              | NVT              |
| 2                               | Geen inspecties uitgevoerd                                                        | 1/3            | NVT              |
| 3                               | Aantoonbare inspecties/reviews op ontwerpen en code uitgevoerd                    | 0              | 1/3              |
| 4                               | Aantoonbaar Fagan inspecties uitgevoerd op alle ontwerpen, code en testdocumenten | -1/2           | 0                |

#### *Hoeveelheid wijzigingen van de bouwsteen*

Een ontwerp dat regelmatig aan grootschalige wijzigingen blootgesteld wordt, heeft een grotere kans om fouten te bevatten doordat wijzigingen niet geheel doordacht of onvolledig worden doorgevoerd. Ook neemt de kans toe dat wijzigingen in ontwerpdocumenten wel worden doorgevoerd, maar in daaruit voortvloeiende documenten niet (geheel) worden doorgevoerd.

| <b>3 Hoeveelheid wijzigingen ten opzichte originele ontwerp/eisenpakket van de bouwsteen</b> |                                                   |      |
|----------------------------------------------------------------------------------------------|---------------------------------------------------|------|
| 1                                                                                            | Onbekend                                          | 0    |
| 2                                                                                            | Zeer frequente of enkele fundamentele wijzigingen | 2/3  |
| 3                                                                                            | Weinig wijzigingen, met zeer geringe impact       | 0    |
| 4                                                                                            | Geen wijzigingen                                  | -1/3 |

Kern van dit aspect is gezamenlijke omvang van de wijzigingen, en de impact die de wijzigingen hebben op alle daaruit voortvloeiende ontwerp documenten. Doel van dit aspect is echt de dynamiek van significante wijzigingen vast te stellen.

#### *Cultuur en samenwerking*

Cultuur is een niet te onderschatten belangrijk onderdeel van samenwerken en de effectiviteit van die samenwerking. Met name die effectiviteit van de samenwerking heeft ook veel invloed op de kans die een TUB heeft op falen. Voor het scoren van dit aspect is alleen de cultuur in het specifieke ontwikkelteam van de software van belang.

| <b>4 Cultuur en samenwerking</b> |                                  |      |
|----------------------------------|----------------------------------|------|
| 1                                | Onbekend                         | 0    |
| 2                                | Op regels gebaseerde organisatie | 1/3  |
| 3                                | Doelgerichte organisatie         | 0    |
| 4                                | Zelflerende organisatie          | -1/2 |

Deze typering is gebaseerd op de definitie van cultuur en samenwerking die in de IAEA-TECDOC-1329 is gedefinieerd.

#### *Opleidingsniveau en ervaring ontwikkelteam*

Ervaring met het domein is een belangrijke voorwaarde voor het doorgronden van de requirements, en het voorkomen van veelgemaakte fouten in een specifiek domein.

| <b>5 Opleidingsniveau en ervaring ontwikkelteam</b> |                                                                                                           |      |
|-----------------------------------------------------|-----------------------------------------------------------------------------------------------------------|------|
| 1                                                   | Onbekend                                                                                                  | 0    |
| 2                                                   | Geen kennis met systeemontwikkeling voor het specifieke domein (onbewust onbekwaam)                       | 1    |
| 3                                                   | Te weinig kennis met systeemontwikkeling voor het specifieke domein (bewust onbekwaam)                    | 1/2  |
| 4                                                   | Gewenste kennis met systeemontwikkeling voor het specifieke domein (bewust bekwaam)                       | 0    |
| 5                                                   | Uitstekende kennis en veel ervaring met systeemontwikkeling voor het specifieke domein (onbewust bekwaam) | -1/2 |

#### *Samenwerking met de opdrachtgever*

Het achterliggende reden van dit aspect is dat sub-optimale oplossingen van nature onbetrouwbaarder zijn doordat:

- een opdrachtgever niet snapt wat hij moet bestellen bij de software-ontwikkelaar;
- er slechte ontwerpbeslissingen genomen worden doordat de belangen van software-ontwikkeling als ondergeschikt wordt beschouwd aan andere disciplines.

| <b>6 Samenwerking met Opdrachtgever</b> |                                                                                                               |     |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------|-----|
| 1                                       | Onbekend.                                                                                                     | 0   |
| 2                                       | Niet nauw betrokken opdrachtgever met weinig IT kennis, sterk contractueel/financieel gedreven opdrachtgever. | 1/2 |
| 3                                       | Zijdelings betrokken opdrachtgever met matige IT kennis.                                                      | 0   |

|   |                                                                                                                                                                                                                                                                                            |      |
|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|
| 4 | Sterk betrokken opdrachtgever met voldoende kennis en open dialoog waarbij de opdrachtgever bereid is om overall architectuur te wijzigen als dat de software betrouwbaarheid ten goede komt. Er is sprake van een systems engineering aanpak voor de gehele ontwikkeling van het systeem. | -1/2 |
|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|

## 6.2

### Software product

#### *Complexiteit beslissingslogica van de TUB*

Complexe software is zeer moeilijk betrouwbaar te krijgen. Complexiteit heeft drie belangrijke effecten die de betrouwbaarheid van de TUB nadelig beïnvloeden:

- Complexe software bevat veel paden, waardoor eenvoudig fouten gemaakt kunnen worden door gebrek aan overzicht en inzicht van de exacte werking van de software;
- Complexe software is moeilijk te overzien, waardoor modificatie en bug-fixing veel vaker resulteert in nadelige neven-effecten;
- Complexe software is zeer moeilijk testbaar, waardoor de effectiviteit van testen significant afneemt.

| 7 Complexiteit beslissingslogica van de specifieke TUB |                                                                                                                          |      |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|------|
| 1                                                      | Onbekend.                                                                                                                | 0    |
| 2                                                      | Beslislogica is zeer complex (bevat veel vertakkingen en uitzonderingen), McCabe index groter dan 60.                    | 1/2  |
| 3                                                      | Beslislogica matig complex (bevat enkele uitzonderingssituaties), McCabe index tussen 30 en 60.                          | 0    |
| 4                                                      | Beslislogica is redelijk eenvoudig (bevat enkele zeer geïsoleerde uitzonderingssituaties), McCabe index tussen 10 en 30. | -1/3 |
| 5                                                      | Beslislogica en foutherkenning zijn erg eenvoudig, McCabe Index kleiner dan 10.                                          | -1/2 |

#### *Omvang TUB*

De omvang van de software is mede bepalend voor de faalkans. Niet alleen omdat statistisch de kans op fouten toeneemt als de code groter wordt, maar ook omdat de overzichtelijkheid en begrijpelijkheid afneemt en daardoor sneller fouten ontstaan.

| 8 Omvang TUB (Lines of code) |                         |      |
|------------------------------|-------------------------|------|
| 1                            | Onbekend.               | 0    |
| 2                            | Meer dan 50.000         | 1/2  |
| 3                            | Tussen 10.000 en 50.000 | 1/3  |
| 4                            | Tussen 5.000 en 10.000  | 0    |
| 5                            | Tussen 1.000 en 5.000   | -1/3 |
| 6                            | Minder dan 1000         | -1/2 |

#### *Helderheid van gebruikte architectuurconcepten*

Een goede architectuur is een belangrijke voorwaarde om software goed begrijpbaar, bouwbaar, testbaar en onderhoudbaar te krijgen. Het andere extreem is slecht georganiseerde code (ook wel spaghetti-code genaamd) waar geen heldere afbakening van verantwoordelijkheden benoemd zijn, waardoor het resultaat moeilijk te begrijpen en zeer moeilijk te testen is.

| 9 Helderheid gebruikte architectuurconcepten |
|----------------------------------------------|
|----------------------------------------------|

|   |                                                                                                                                                                                                                                                         |                |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| 1 | Onbekend.                                                                                                                                                                                                                                               | 0              |
| 2 | Geen heldere afbakening TUBs in ontwerp benoemd.                                                                                                                                                                                                        | $\frac{1}{2}$  |
| 3 | Wel taakuitvoering op hoofdlijnen benoemd, maar geen navolging gegeven in ontwikkeling.                                                                                                                                                                 | $\frac{1}{3}$  |
| 4 | Er is een scheiding van taakuitvoering tussen TUBs beschreven, welke het principe van "maximale cohesie en minimale koppeling respecteert", maar deze is passief bewaakt tijdens het ontwikkelproces.                                                   | 0              |
| 5 | Er is een scherpe scheiding van taakuitvoering tussen TUBs beschreven op basis van geldende documenten, welke het principe van "maximale cohesie en minimale koppeling respecteert", en deze is aantoonbaar actief bewaakt tijdens het ontwikkelproces. | $-\frac{1}{2}$ |

#### *Gebruik van een certified compiler*

Compilers hebben een belangrijke invloed op de kwaliteit van software. De meest voor de hand liggende eigenschap van een compiler is dat deze een betrouwbare manier moet zijn om programmacode om te zetten in executeerbare machinecode. Maar compilers hebben ook kwalitatieve eigenschappen, zoals expliciete beperkingen in de constructies gebruikt in programmacode.

| <b>10 Gebruik van een certified compiler</b> |                                                                                                                                                                                                                                 | <b>Normaal</b> | <b>SIL3/SIL4</b> |
|----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|------------------|
| 1                                            | Onbekend.                                                                                                                                                                                                                       | 0              | NVT              |
| 2                                            | Gebruik van een willekeurige compiler.                                                                                                                                                                                          | $\frac{1}{3}$  | NVT              |
| 3                                            | Gebruik van een compiler waar de ontwikkelaar langdurige ervaring mee heeft.                                                                                                                                                    | 0              | $\frac{1}{3}$    |
| 4                                            | Gebruik van een certified compiler in combinatie met een gevalideerde safe subset                                                                                                                                               | $-\frac{1}{2}$ | 0                |
| 5                                            | Gebruik van een certified compiler in combinatie met een gevalideerde safe subset met bijbehorende kalibratiesets en testprotocol om compiler te ijken/testen, wat voor elke nieuwe versie van de compiler structureel gebeurt. | $-\frac{2}{3}$ | $-\frac{1}{3}$   |

Er wordt hier onderscheid gemaakt tussen de normale en SIL3/4 situatie omdat op hogere SIL-niveau's het gebruik van een certified compiler verplicht is.

## **6.3**

### **Requirements traceability/verifieerbaarheid**

#### *Traceerbaarheid van requirements*

Traceerbaarheid van requirements garandeert dat eisen ook expliciet worden geïmplementeerd en getest. Hiermee wordt aangetoond dat een TUB "doet wat hij moet doen".

| <b>11 Traceerbaarheid van eisen door het proces heen</b> |                                                                                                  | <b>Normaal</b> | <b>SIL3/SIL4</b> |
|----------------------------------------------------------|--------------------------------------------------------------------------------------------------|----------------|------------------|
| 1                                                        | Onbekend                                                                                         | 0              | NVT              |
| 2                                                        | Geen traceerbaarheid                                                                             | $\frac{1}{3}$  | NVT              |
| 3                                                        | Aantoonbaar traceerbaar naar testscripts                                                         | 0              | NVT              |
| 4                                                        | Aantoonbaar traceerbaar naar architectuur en testen                                              | $-\frac{1}{3}$ | $\frac{1}{3}$    |
| 5                                                        | Aantoonbaar traceerbaar van veiligheidskritische eisen tot aan de code en individuele testen toe | $-\frac{2}{3}$ | 0                |

|   |                                                                |    |                 |
|---|----------------------------------------------------------------|----|-----------------|
| 6 | Aantoonbare volledige traceerbaarheid                          | -1 | - $\frac{1}{3}$ |
| 7 | Aantoonbaar wiskundig/logisch bewezen correcte traceerbaarheid | -2 | - $\frac{1}{2}$ |

Er wordt hier onderscheid gemaakt tussen de normale en SIL3/4 situatie omdat op hogere SIL-niveau's het gebruik requirements traceerbaarheid in praktijk verplicht is.

## 6.4 Testen

### *Testtechnieken en dekkingsgraad*

Testen hebben een grote invloed op de uiteindelijke betrouwbaarheid van de bouwsteen. Met name de diepgang en dekkingsgraad van testen zijn van belang. Expliciet moet worden opgemerkt dat de dekkingsgraad gemeten wordt ten opzichte van de af te schatten bouwsteen.

| 12 Testtechnieken en dekkingsgraad van de TUB |                                                                                           | Normaal          | SIL3/SIL4       |
|-----------------------------------------------|-------------------------------------------------------------------------------------------|------------------|-----------------|
| 1                                             | Onbekend                                                                                  | 0                | NVT             |
| 2                                             | Geen gedocumenteerde testen uitgevoerd                                                    | 0                | NVT             |
| 3                                             | Wel testen gedocumenteerd, geen formele testtechnieken gehanteerd; Dekkingsgraad onbekend | - $\frac{1}{3}$  | NVT             |
| 4                                             | Formele testtechniek(en) gehanteerd met lage dekkingsgraad                                | - $\frac{1}{2}$  | $\frac{2}{3}$   |
| 5                                             | Formele testtechniek(en) gehanteerd met gemiddelde dekkingsgraad                          | - $\frac{2}{3}$  | $\frac{1}{2}$   |
| 6                                             | Formele testtechniek(en) gehanteerd met hoge dekkingsgraad                                | -1               | 0               |
| 7                                             | Formele testtechniek(en) gehanteerd met aantoonbare (gemeten) hoge dekkingsgraad          | -1 $\frac{1}{3}$ | - $\frac{1}{3}$ |

Er wordt hier onderscheid gemaakt tussen de normale en SIL3/4 situatie omdat

- op hogere SIL-niveau's een formele testtechniek verplicht is.
- door de natuur van de IEC 61508 er sprake is van verminderde meeropbrengsten op deze SIL-niveaus: reviews en modelverificaties ontdekken meer fouten vooraf dan testen dat doen.

## 6.5 Executieomgeving/gebruik

### *Multiprocesomgeving*

Multiproces-omgevingen leiden tot concurrerende processen voor dezelfde resources. Hierdoor wordt met name tijdigheid, maar soms ook stabiliteit, veel minder sterk te garanderen.

| 13 Multiprocesomgeving |                                                                                    |                 |
|------------------------|------------------------------------------------------------------------------------|-----------------|
| 1                      | Onbekend                                                                           | 0               |
| 2                      | Meerdere TUBs in een gevirtualiseerde omgeving                                     | $\frac{1}{2}$   |
| 3                      | Meerdere TUBs die onafhankelijk van elkaar op één stuk hardware worden uitgevoerd. | $\frac{1}{3}$   |
| 4                      | Maximaal één TUB op een dedicated OS met een dedicated CPU                         | 0               |
| 5                      | Eén TUB op een dedicated CPU en memory op geen of een triviaal OS                  | - $\frac{1}{3}$ |

*Aanwezigheid van representatieve velddata taakuitvoering*

Velddata toont aan dat de software ook daadwerkelijk goed functioneert.

| <b>14 Aanwezigheid representatieve velddata gedurende taakuitvoering</b> |                                                                                                                       | <b>Normaal</b> | <b>SIL3/SIL4</b> |
|--------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|----------------|------------------|
| 1                                                                        | Onbekend                                                                                                              | 0              | NVT              |
| 2                                                                        | Geen velddata aanwezig, zelfs niet uit (schaduw)draaien van de TUB                                                    | $\frac{1}{3}$  | $\frac{1}{3}$    |
| 3                                                                        | Beperkte gegevens aanwezig en geanalyseerd uit periode tijdens uitvoeren van de functie van de TUB                    | 0              | 0                |
| 4                                                                        | Significante hoeveelheid gegevens aanwezig en gebruikt uit periode tijdens uitvoeren van de functie van de TUB        | -1             | $-\frac{1}{3}$   |
| 5                                                                        | Veel representatieve velddata aanwezig en gebruikt van identieke of sterk vergelijkbare toepassingen met dezelfde TUB | -2             | $-\frac{1}{2}$   |

Er wordt hier onderscheid gemaakt tussen de normale en SIL3/4 situatie omdat er sprake is van verminderde meeropbrengsten op deze SIL-niveaus: reviews, modelverificaties en testen ontdekken meer fouten vooraf, waardoor het netto effect van draaiuren minder rendement oplevert.

#### *Monitoring*

Monitoring houdt de TUB in de gaten tijdens de uitvoering van zijn taak. In gemonitord systeem worden fouten gedetecteerd als ze optreden, terwijl in een ongemonitord systeem fouten ongemerkt aanwezig kunnen blijven.

| <b>15 Monitoring van de TUB</b> |                                                           |                |
|---------------------------------|-----------------------------------------------------------|----------------|
| 1                               | Onbekend                                                  | 0              |
| 2                               | Geen aanwezig                                             | $\frac{1}{3}$  |
| 3                               | Weinig/kort gemonitord gedurende taakuitvoering           | 0              |
| 4                               | Langdurige monitoring, maar niet frequente taakuitvoering | $-\frac{1}{3}$ |
| 5                               | Langdurige/frequente monitoring tijdens taakuitvoering    | $-\frac{1}{2}$ |

## 7 Uitkomsten onderhoudsslagen

### 7.1 Eerste onderhoudsslag (2008-2017)

Er is door Rijkswaterstaat een onafhankelijke review op de TOPAAS aanpak uitgevoerd door Prof. Dr. J.A. Bergstra, Dr. M.I.A. Stoelinga en Prof. Dr. J.C. van Vliet. Hieronder worden opmerkingen uit dit review behandeld.

Het doel van deze eerste onderhoudsslag is om consistentie van teksten en toelichtingen te verbeteren, om hiermee de consistentie van analyses te verhogen. Aanpassingen aan de aspecten en/of de scoring vergt nader onderzoek.

De commentaarlijst van alle bij Rijkswaterstaat bekende reviews is in beheer bij Rijkswaterstaat.

#### *Onderbelichting Testproces*

De reviewers zijn van mening dat het testproces onderbelicht is, en dat testen meer aandacht verdient. De expertgroep ziet echter dat het testproces al vertegenwoordigd is in:

- Traceerbaarheid van requirements (Aspect 11);
- Dekkingsgraad van testen (Aspect 12);
- Velddata (Aspect 14).

Uit de verdere analyse van de expertgroep blijkt ook dat deze invloed zeer groot is: in totaal kan men met grondig testen vijf ordegroottes verschil maken op de uitkomst. Dit is in praktijk al een zeer dominante factor. In de toelichtende tekst bij deze aspecten is meer aandacht voor het testproces opgenomen.

Naast de mogelijke oververtegenwoordiging van testen, is de expertgroep van mening dat:

- TOPAAS een output-georiënteerde methode is: kijken naar randvoorwaarden zoals testplannen past niet binnen die filosofie. TOPAAS kijkt bijvoorbeeld ook niet naar engineeringplannen en coding standards.
- De huidige metingen de kern van testen raken: de breedte (aspect 11) en de diepte (Aspecten 12 en 14).

#### *De aanwezigheid van perverse prikkels in aspect 1*

De reviewers merken op dat:

*Met de huidige gewichten kan het zou zijn dat project ontwikkeld met SIL2 en extra eisen die voor SIL3 vereist worden betrouwbaarder scoort dan een project ontwikkeld met SIL3. Dit betekent dat de impact van de SIL-levels (aspect 1) te klein en het onderscheid tussen SIL3/4 en normaal (aspecten 10 t/m 15) te groot is.*

De TOPAAS expertgroep erkent het probleem dat er “perverse prikkels” in de scoren van aspect 1 aanwezig zijn in combinatie met de daarop volgende aspecten (2, 10-12 en 14).

De analyse van de expertgroep constateert dat dit een zeer diepgeworteld probleem is dat voortkomt uit het gebruik van de IEC 61508 als meetinstrument. De SIL-classificatie van de IEC 61508 suggereert een lineaire relatie tussen SIL-niveau en de effecten van de maatregelen. De praktijk is echter anders: SIL 1 en 2 zijn onderling zeer sterk gerelateerd, net als SIL 3 en 4. Men kan stellen dat vanuit de

vereiste maatregelen voor software ontwikkeling gezien de IEC 61508 eigenlijk maar twee modi kent: SIL1/2 en SIL3/4.

Hiermee is aspect 1 tot een vrij bot instrument geworden: er wordt fijnmaziger gescoord dan het onderscheid in onderliggende maatregelen eigenlijk rechtvaardigen. Probleem is echter dat het SIL-niveau en de IEC 61508 (en de industrie specifieke varianten daarvan) onderdeel uitmaken van nationale en internationale wet- en regelgeving, en ook onderdeel is van de Nederlandse rechtspraak.

Direct resultaat is dat deze sprong in maatregelen zich moeilijk laat corrigeren. Een gevoeligheidsanalyse van de scoring laat ook zien dat het probleem blijft bestaan ongeacht de wijziging in scoring, zelfs als men voor alle 4 de SIL-levels aparte scoring toevoegt is dit niet te beheersen zonder dat de kalibratie van het model verloren gaat.

De expertgroep is wel van mening dat dit opgelost moet worden. Hiervoor zijn twee sporen uitgezet:

- De versie 2.0 van TOPAAS bevat in de handleiding een expliciete aanwijzing om deze perverse prikkel te voorkomen. Als men een SIL-3 proces uitvoert mag men niet om rekenkundige redenen SIL-2 kiezen.
- Er wordt voor versie 3.0 van TOPAAS een wijziging van Aspect 1 voorgesteld die zich meer richt op de aanwezigheid van een proces, maar dat minder hangt aan de IEC61508 (met al zijn "dubbeltellingen").

#### *Omgang kennis*

De reviewers merken op:

*Er worden hier diverse zaken [in aspect 5] gemeten: opleidingsniveau van de ontwikkelaars, ervaring met software, en domeinkennis. Dit zou uit elkaar getrokken moeten worden, en de affiniteit met software zou benadrukt moeten worden.*

De expertgroep onderkent dat dit aspect aangescherpt moet worden in TOPAAS 2. De term "affiniteit met software" is echter te ambigu en legt de lat te laag. De expertgroep is van mening dat:

- het belangrijker is om inhoudelijke kennis en begrip te hebben van het specifieke domein, en dat dit aantoonbaar is door werkervaring in het domein;
- men die werkervaring ook in dezelfde rol actief heeft moeten vervullen. Dit heeft het praktische gevolg dat men voor een tunnel verwacht dat de programmeur niet alleen al x jaar programmeert, maar dat ook doet in tunnelprojecten.

Dit is aanleiding geweest de toelichting op het aspect in TOPAAS 2 (§4.3.5 van de handleiding) aan te scherpen.

#### *Velddata*

De reviewers merken op:

*Dit aspect vinden wij onduidelijk: wat zijn velddata, en hoe kom je daaraan, voor ingebruikname van de software? Wat zijn schaduwdraaien van het systeem?*

Dit is een geluid dat ook in het veld wordt herkend. De betekenis van velddata is nader aangescherpt om verwarring te voorkomen. In TOPAAS 2 wordt explicieter onderscheid gemaakt tussen testdata en velddata. Dit is inhoudelijk verder uitgewerkt in §4.3.14 van de handleiding TOPAAS 2.

### *Hoe om te gaan met faalfrequentie?*

Centraal in TOPAAS is het concept taakuitvoering, wat de betrouwbaarheid per aanspraak (Q) in zich heeft. Een faalfrequentie ( $\lambda$ ) geeft de kans van onbeschikbaarheid per tijdseenheid weer. Dit zijn te relateren aspecten, maar in praktijk zitten hier bij software zeer lastige vraagstukken achter. Theoretisch gezien, vanuit de reliability engineering gezien, zitten hier drie concepten achter:

- De aanspreekfrequentie
- De missieduur
- De reparatietijd

De aanspreekfrequentie van een TUB is niet eenvoudig te bepalen: spreekt een gelijklooplegeling nu één keer per aanspraak aan, of elke seconde een maal met nieuwe input? Het risico dat men teveel extrapoleert op conservatieve waarden is extreem groot: als men uitgaat van een gezond systeem met een faalkans  $10^{-4}$  op aanspraak dat per dag 10.000 keer wordt aangesproken zal daarmee bijna een garantie van falen opleveren. Dit is geen realistische uitkomst en derhalve ook geen triviale berekening.

De missieduur kent een vergelijkbaar probleem: er zijn oplossingen bekend die een zeer lange missieduur kennen, maar die in praktijk geïmplementeerd zijn door een softwarematige sleep()-operatie van tientallen minuten. In essentie wordt de missieduur van software dus niet in doorlooptijd gemeten, maar in termen van het aantal beslissingen dat het moet nemen om het gestelde doel te bereiken. Bij elke beslissing wordt in veel gevallen dezelfde beslissing herhaald (moet ik de klep blijven sluiten?), waar in andere gevallen echt andere beslissingen genomen worden (wat moet het setpoint van de klep worden?), en soms worden er zelfs zaken modelmatig geëxtrapoleerd (volgens mijn model zal de waterstand over vijf minuten op niveau x zijn, dus moet ik nu de klep sluiten?), waarbij het faalgedrag radicaal anders is. Dit zijn geavanceerde analyses die TOPAAS ontstijgen.

Reparatietijd moet zich richten op functioneel herstel, maar zelfs dan is het lastig in te schatten. Waar in sommige gevallen/oplossingen het eenvoudig is om een PLC (volgens een gevalideerde werkinstructie) te resetten en dan het proces zichzelf te laten initialiseren, moet in andere gevallen het proces handmatig naar een bekende toestand worden gebracht voordat de PLC het kan overnemen of moet er zelfs een softwarematige aanpassing worden aangebracht voordat men het proces kan hervatten. Hierbij kan dus de reparatietijd variëren van enkele secondes tot enkele uren en soms zelfs weken. Hier kunnen ontwerpkeuzes helpen (systematisch gebruik absolute encoders én gedegen redundantie), maar het kwantificeren ervan blijft moeilijk, zeker als een softwarematige aanpassing noodzakelijk kan zijn om extremere (valide) input toch goed af te handelen.

## **7.2**

### **Aandachtspunten**

Voor toekomstige releases van TOPAAS zijn de volgende aandachtspunten nu alvast benoemd. Deze lijst is niet limitatief, maar dit zijn de bekende punten, die middels toelichtingen hun weg in de handleiding hebben gevonden.

#### *Behandeling hoogfrequente en continue systemen*

De referentiedatabase heeft nu nog typische 1x/jaar systemen in zich.

Toevoegingen van tunnels, bruggen en continue systemen uit de analyses van afgelopen en komende periode dienen onderzocht te worden of dit een welkome aanvulling gaat geven.

#### *Externe model eigenschappen*

Inbreng van velddata vs. Bayesiaans updaten: een systeem dat al meerdere jaren zonder problemen draait kan gemodelleerd worden met een a-prio kans uit TOPAAS (situatie terug in de tijd) met Bayesiaans updaten, of een a-prio kans (situatie nu) met inbreng van velddata. Gevoeligheid voor deze keuze dient onderzocht te worden.

#### *Interne model consistenties*

Een modelmatige inconsistentie is de SIL-kwestie. Door op een lager SIL-niveau te score, maar wel alle aanvullende maatregelen te treffen is de score anders dan het juiste SIL-niveau aan te geven en geen aanvullende punten te krijgen voor de juiste maatregelen.

Een ander punt is dat onderzoek moet worden gedaan of er significante verschillen zijn als TUB's gesplitst of samengevoegd worden. Ofwel dat door de kwalitatieve analyse de resultaten worden beïnvloed.

## 8 Vergelijking met de referentiedatabase

Ter controle is er een referentiedatabase opgezet. Deze referentiedatabase dient ter controle van de scoring van de TOPAAS checklist. Doel hiervan is het monitoren van de relatie tussen het expert-oordeel en de modelmatige uitkomst van TOPAAS. Hierbij zijn dus alle projecten onafhankelijk gescoord van de TOPAAS invulling voor een lage aanspreekfrequentie (gemiddeld één keer per jaar).

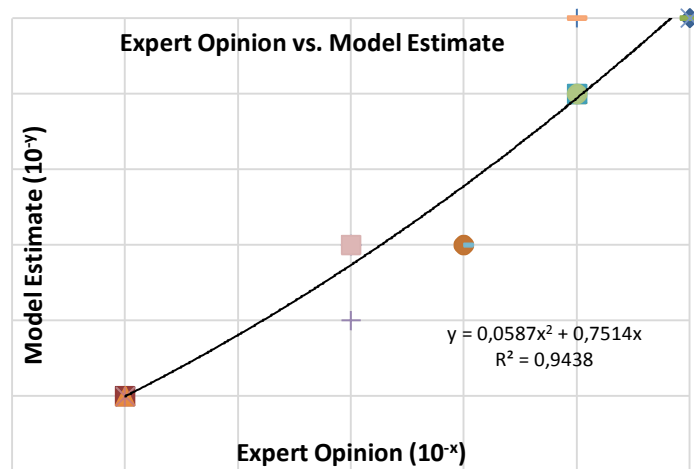
### 8.1 Referentiedatabase

In totaal zijn er, naast de zeven archetypen, ook 13 echte projecten ingevoerd. Allen uit een veiligheidskritische/mission critical omgeving met een sterke verbinding met de besturing van elektromechanische componenten. In de benodigde faalkans voor elke component zat een sterke variatie, alsmede ook de inschatting van de experts van de faalkans. Dit had de volgende resultaten:

| Projectnaam                                          | Expert | Model |
|------------------------------------------------------|--------|-------|
| ARCHITYPE - Ultimate White box                       | 5      | 5     |
| ARCHITYPE - Worst White Box                          | 0      | 0     |
| ARCHITYPE - Ultimate Black Box                       | 4      | 4     |
| ARCHITYPE - Worst Black Box                          | 0      | 0     |
| ARCHITYPE - Eigen knutselwerk                        | 0      | 0     |
| ARCHITYPE - Intensief gebruikte COTS component       | 3      | 2     |
| ARCHITYPE - Zeer complex SIL-4 systeem               | 4      | 5     |
| Rijkswaterstaat - INWin Oosterscheldekering          | 5      | 5     |
| Rijkswaterstaat - Proces Oosterscheldekering         | 5      | 5     |
| DNV – BLGS                                           | 2      | 3     |
| DNV – MDP                                            | 4      | 4     |
| DNV – SCADA                                          | 0      | 0     |
| DNV – BEEG                                           | 5      | 5     |
| DNV – VILM                                           | 0      | 0     |
| Rijkswaterstaat - Sbk-BOS Interface (happy flow)     | 4      | 4     |
| Rijkswaterstaat - Sbk-BOS interface (uitzonderingen) | 2      | 1     |
| Rijkswaterstaat – SOBEK                              | 3      | 2     |
| Rijkswaterstaat - MDL50                              | 4      | 5     |
| Rijkswaterstaat - Ramspol 2                          | 2      | 2     |
| Rijkswaterstaat - Ramspol 1                          | 2      | 2     |

### 8.2 Globale analyse

Uit het bovenstaande overzicht blijkt al dat de uitkomsten van het model en de initiële afschatting door experts een sterke relatie met elkaar hebben. Zet men de uitkomst van het model uit tegen de afschatting door de experts voor deze 20 projecten, dan komt men tot de volgende grafiek:



Figuur 1: relatie afschatting door experts en model (N=20)

Uit bovenstaande figuur blijkt dus dat er een behoorlijk sterke relatie zit de Expert Opinion en de afschatting van het model. De correlatie is 0,924.

Dit is een situatie waarin de faalkans naar beneden wordt afgerond door het model. Laat men de afronding naar beneden los (maar beperkt men de faalkans afschatting tot de range van 1 tot 10<sup>-5</sup>), dan daalt de correlatie met de expert-afschatting tot 0,9347. Met andere woorden het model wordt optimistischer ten opzichte van de expert-afschattingen. Dit is ongewenst gedrag voor een model dat juist de expert-afschatting probeert te benaderen. Dit leidt tot de conclusie dat het noodzakelijk is om naar beneden af te ronden om het expert-oordeel te benaderen.

Wel blijkt door de trendlijn dat, ondanks dat het model zeer conservatief inschat, het model iets "optimistischer" is dan de experts.

Ander zorgpunt binnen parametermodellen is dominantie: alhoewel er een zeer grote groep van factoren in ogenschouw genomen wordt, zou het uiteindelijk kunnen zijn dat één kleine groep specifieke factoren altijd de faalkans in zeer grote mate bepaald. Een potentiële kandidaat hiervoor is het SIL-Level, met zijn zeer grote invloed op de faalkans (met een bereik van vier ordegrootten). Daarom is bij "Correlatie met de uitkomst" voor elke vraag de bijdrage van het specifieke antwoord (indien aanwezig) van een project afgezet tegen de faalkans die bepaald is door de expert-opinion. Hiermee is het mogelijk om te verifiëren of er sterk dominante factoren in het parametermodel aanwezig zijn. Kijkt men naar de correlatie van de individuele factoren met het beoogde eindresultaat, dan ziet men dat bij elk aspect:

- de individuele bijdrage van veel factoren niet potentieel dominant te noemen zijn. Dat wil zeggen dat de invloed beperkt is tot maximaal één enkele ordegrootte van de uiteindelijke faalkans, waarbij bij veel factoren zelfs een afvlakking plaatsvindt in de range van 10<sup>-4</sup> tot 10<sup>-5</sup>, hetgeen betekent dat het verschil niet specifiek gemaakt wordt door die factor.
- de individuele bijdrage van enkele grote factoren (te weten "SIL-Level", "opleidingsniveau en ervaring ontwikkelaars" en "aanwezigheid representatieve velddata") dat er een behoorlijke invloed is, maar dat ze niet sterk lineair correleren met de verwachte uitkomst. Zelfs als men deze groepen op een willekeurige wijze combineert dan komt men niet tot een sterk correlerende set.

Dit impliceert dat er niet een dominante (set van) factoren aanwezig is die de faalkans bepaalt, en andere factoren dus overbodig zou maken.

Op basis van deze kalibratie kan men dus concluderen dat:

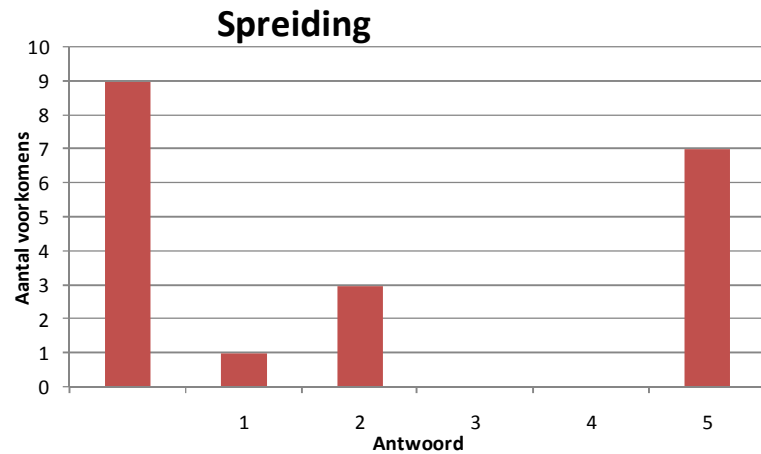
- De uitkomsten van het model een sterke overeenkomst vertonen met de afschatting de faalkans door ervaren experts;
- De factoren allen een redelijke distributie in hun scoring gekend hebben tijdens kalibratie, waardoor de toepasbaarheid in het domein niet beperkt is tot specifieke soorten (veiligheidskritische) systemen in specifieke ontwikkelomgevingen;
- De uitkomsten niet sterk gedomineerd worden door een subset van de factoren, hetgeen in praktijk alle andere factoren buiten die subset overbodig zou maken.

### 8.3

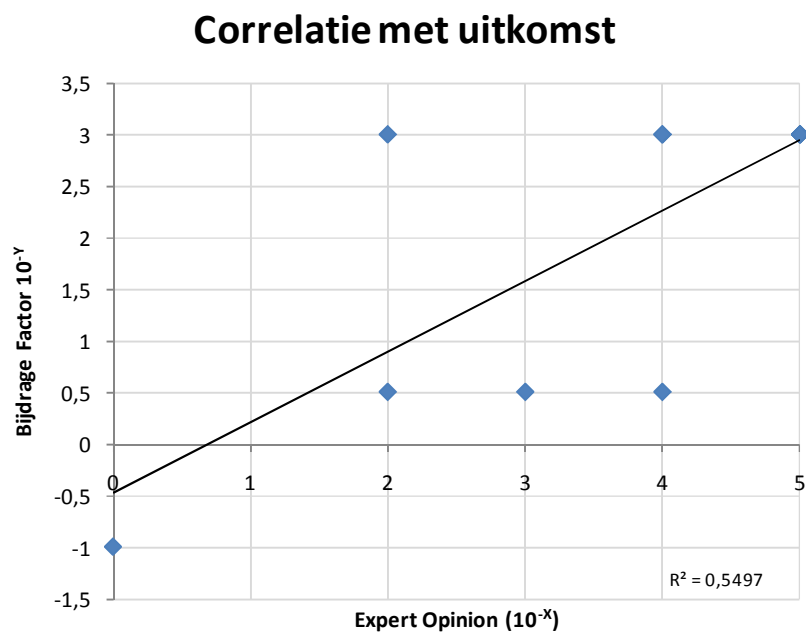
#### **Gedetailleerde analyse**

Bij elke factor zijn onder het kopje "*Spreiding*" staafdiagrammen bijgevoegd die de keuzen zoals die gemaakt zijn door de experts voor de 20 cases. Op de horizontale as staan dus de keuzemogelijkheden zoals die volgen uit de tabellen in hoofdstuk 6 "*Onderbouwing Vragenlijst*", aangevuld met een staaf (de meest linkse) die het lege antwoord (geen scoring) visualiseert. De genoemde cijfers komen dus niet 1-op-1 over op de gehanteerde antwoorden omdat de referentiedatabase opzet het antwoord "*Onbekend*" systematisch niet kent.

De spreiding van de antwoorden bij alle projecten is cruciaal: als de antwoorden bij een specifieke vraag absoluut geen spreiding kennen (altijd hetzelfde ingevuld), dan kan men geen conclusies verbinden aan de uitkomsten van die vraag voor situaties die anders scoren. Als er enige vorm van spreiding is, met name op de uitersten van de scoring, dan kan men in ieder geval nog uitgaan van interpolatie. Maar in alle gevallen is voor de validiteit van scoring ten opzichte van de kalibratie voor alle vragen een significante spreiding noodzakelijk. Zoals de bijlage aangeeft is er door de projecten echt verschillend gescoord, hetgeen aangeeft dat er niet één enkel type project gebruikt is voor de kalibratie, waardoor de scoring redelijk gegeneraliseerd mag worden naar projecten binnen hetzelfde domein.

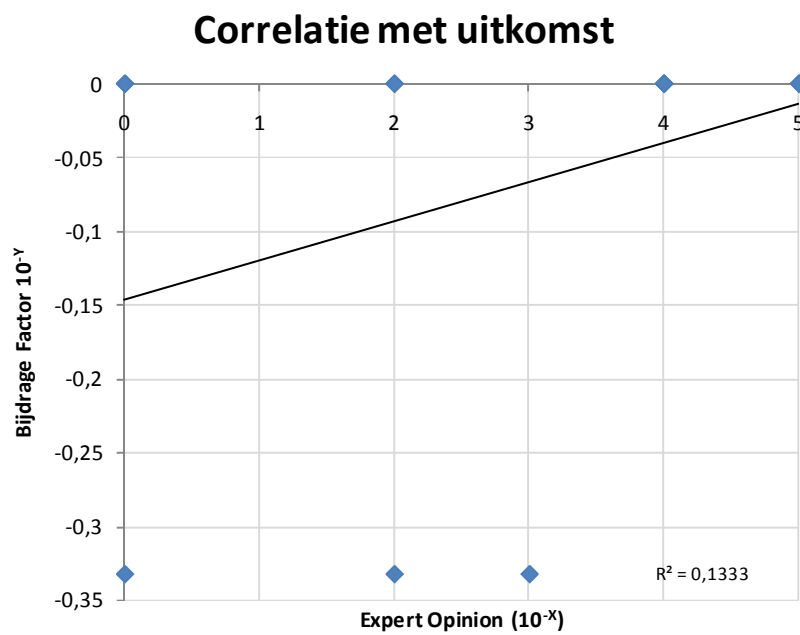
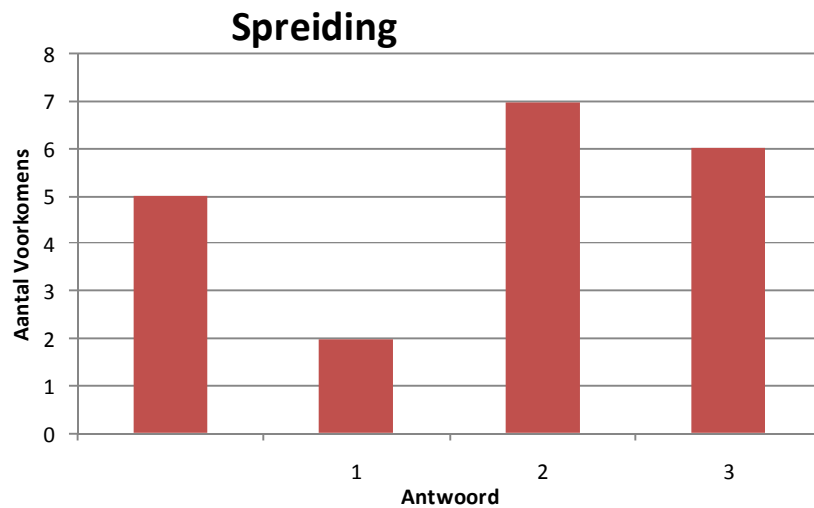
*SIL-level ontwikkelproces*

Wat blijkt uit deze grafiek is dat er een redelijke spreiding in de antwoorden aanwezig: negen projecten zijn niet onderhevig geweest aan een specifiek SIL-niveau (geen antwoord gegeven), en zeven projecten zijn uitgevoerd onder een SIL-4 niveau (antwoord 5).



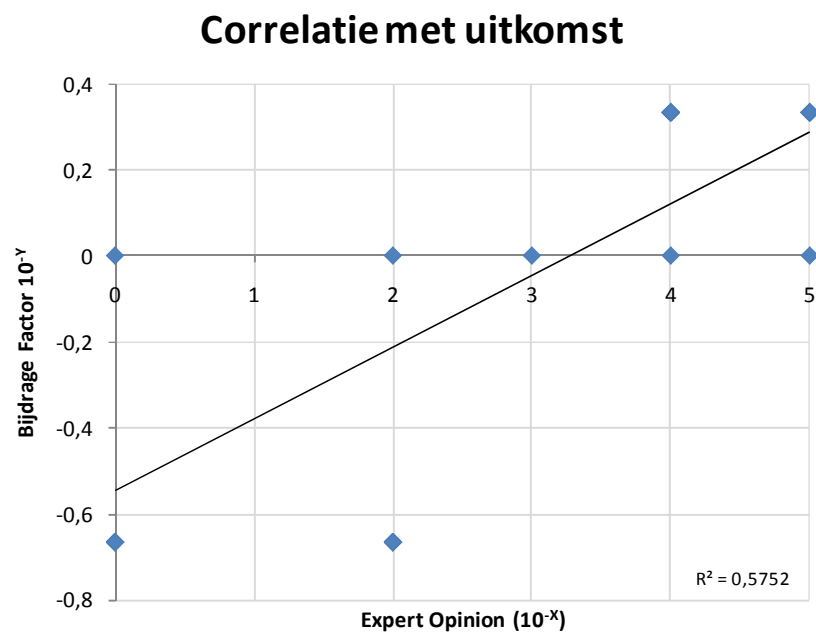
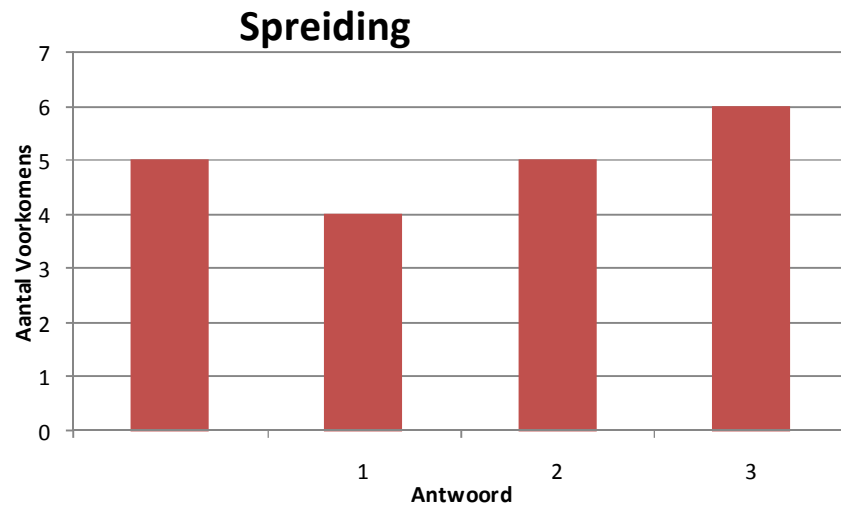
N=11

## *Uitvoeren van Inspecties*

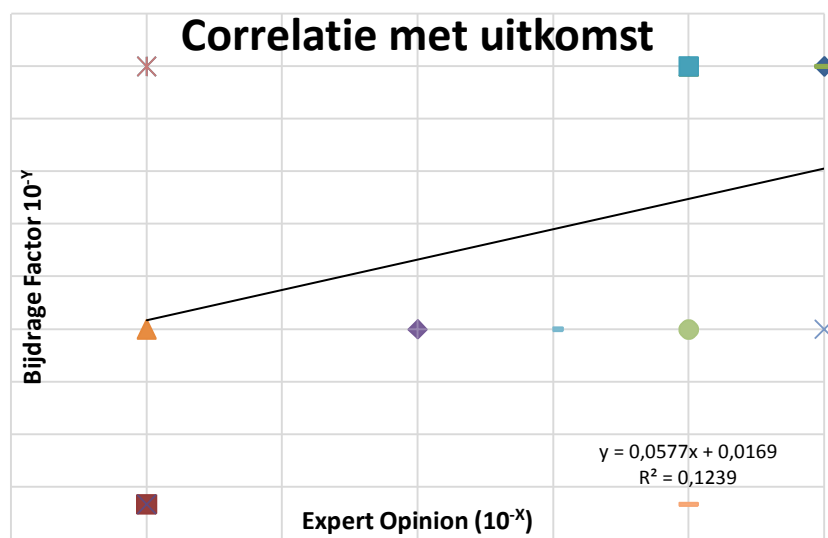
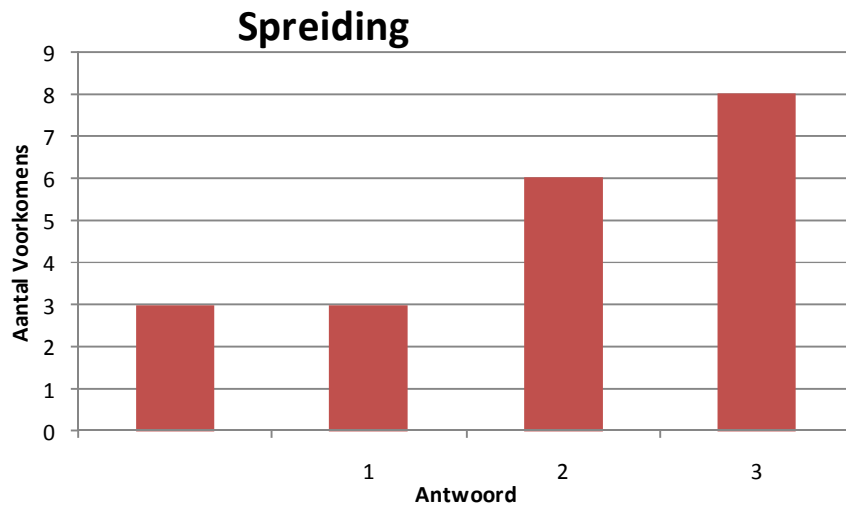


N=15

*Frequentie en impact wijzigingen*

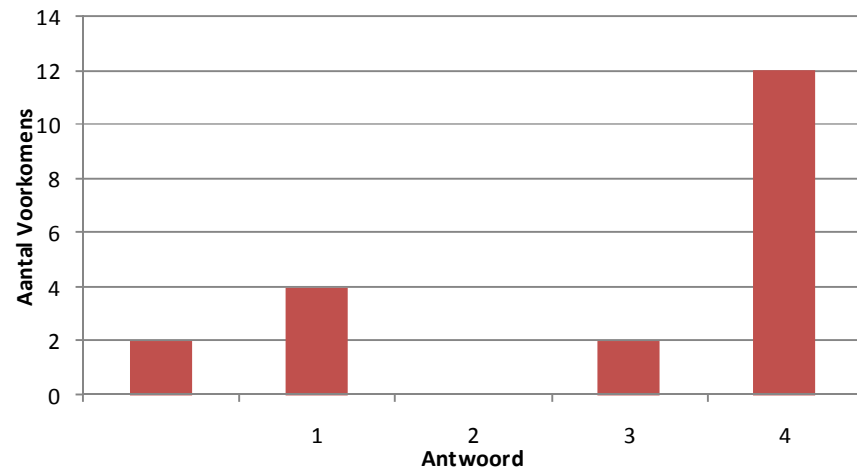


N=15

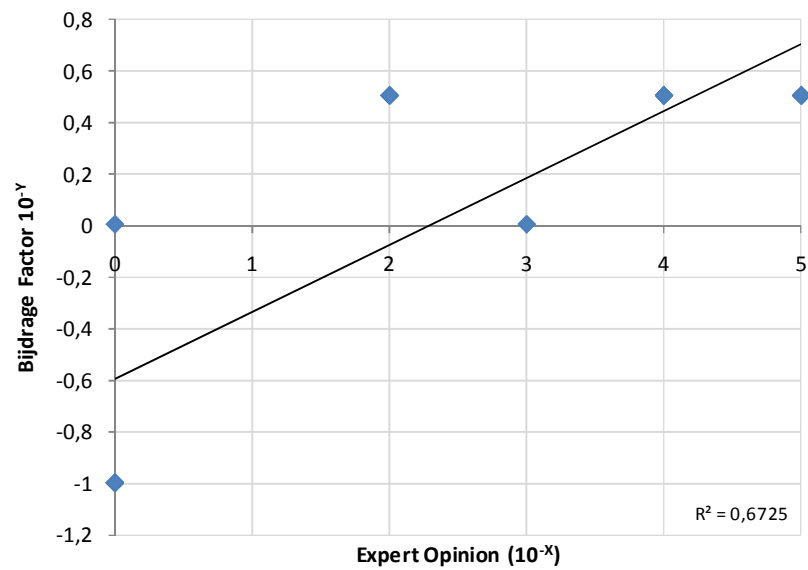


N=17

### Ervaring en opleidingsniveau ontwikkelaars

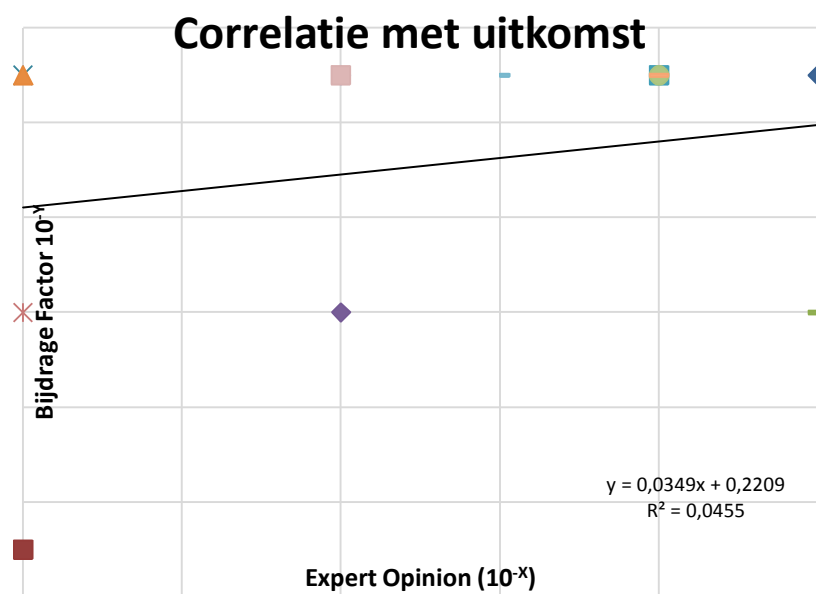
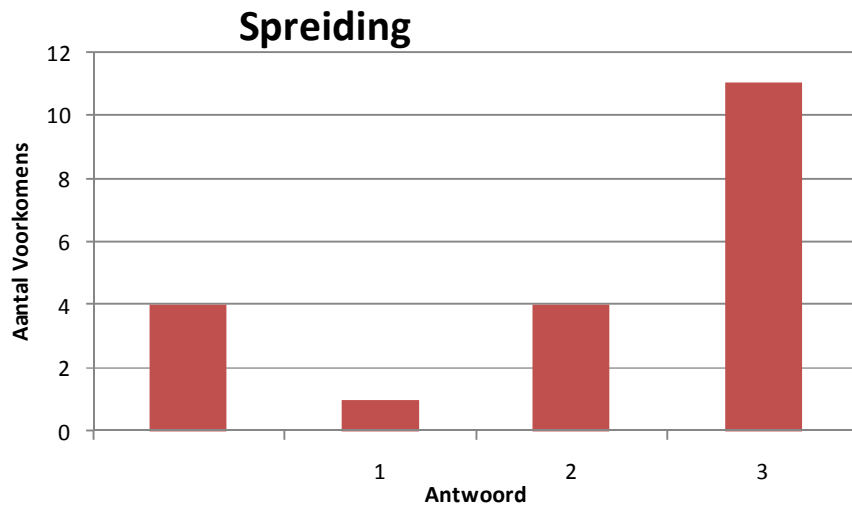


### Correlatie met uitkomst



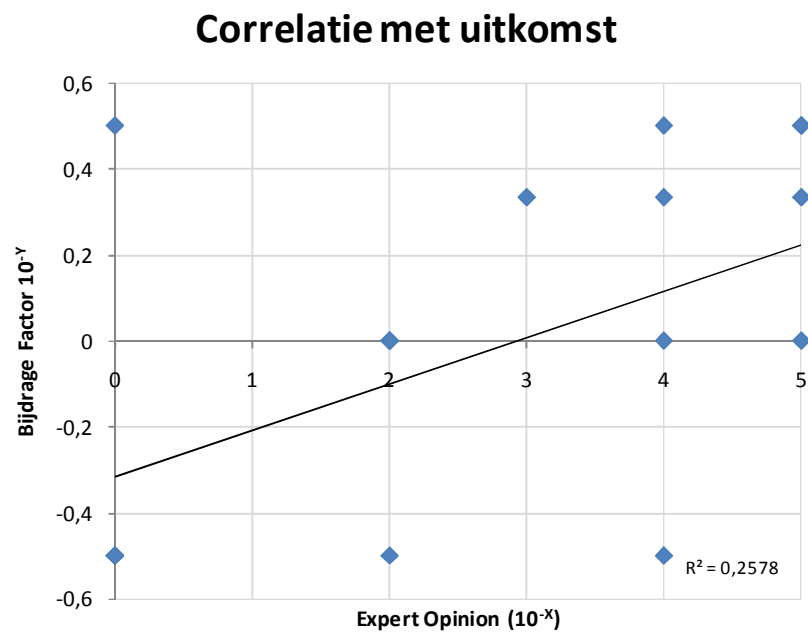
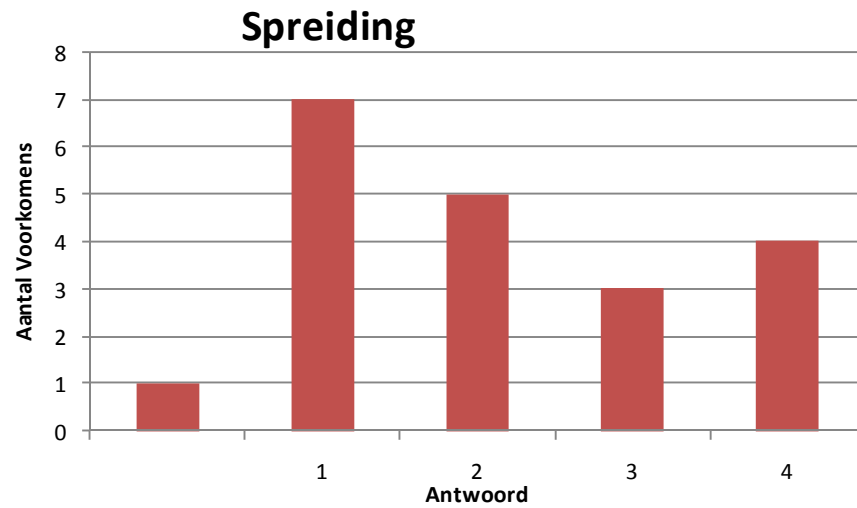
N=18

Relatie Opdrachtgever

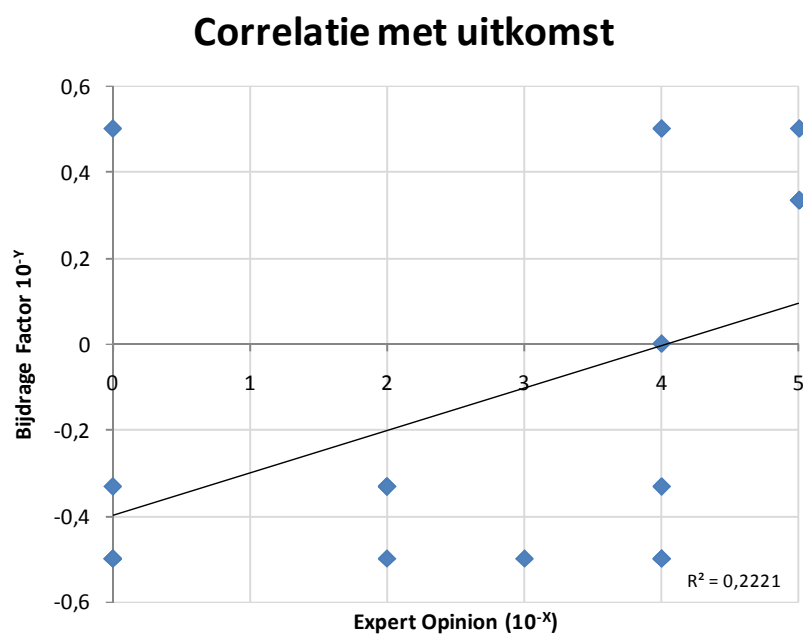
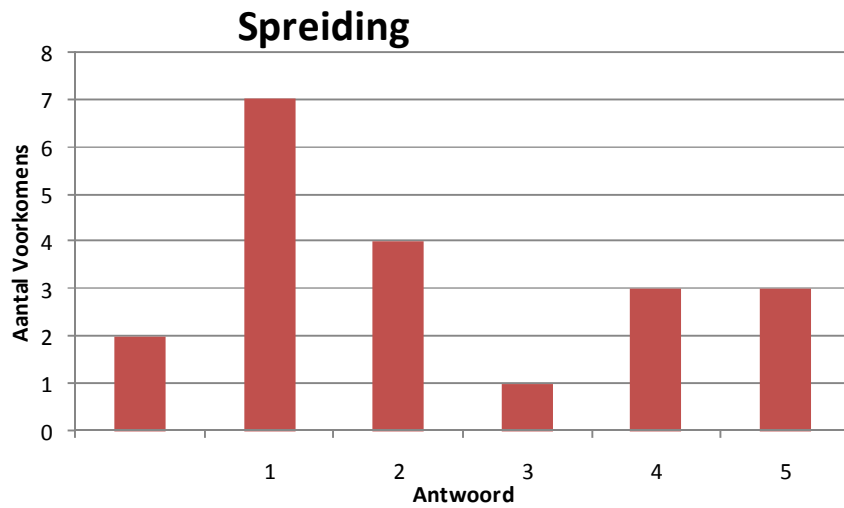


N=16

### Complexiteit oplossing

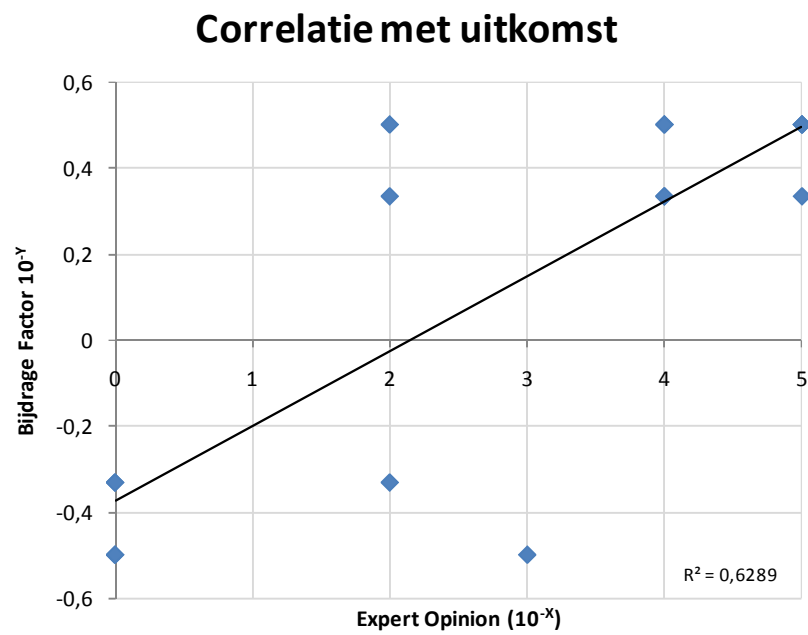
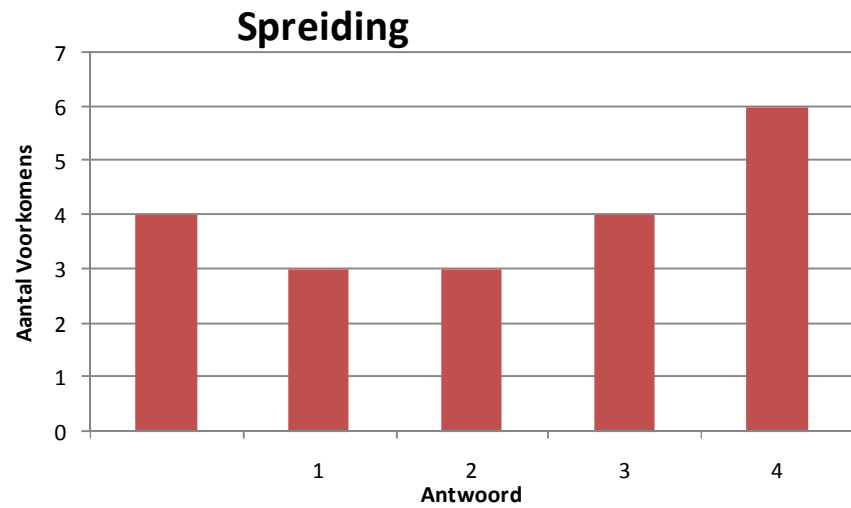


N=19



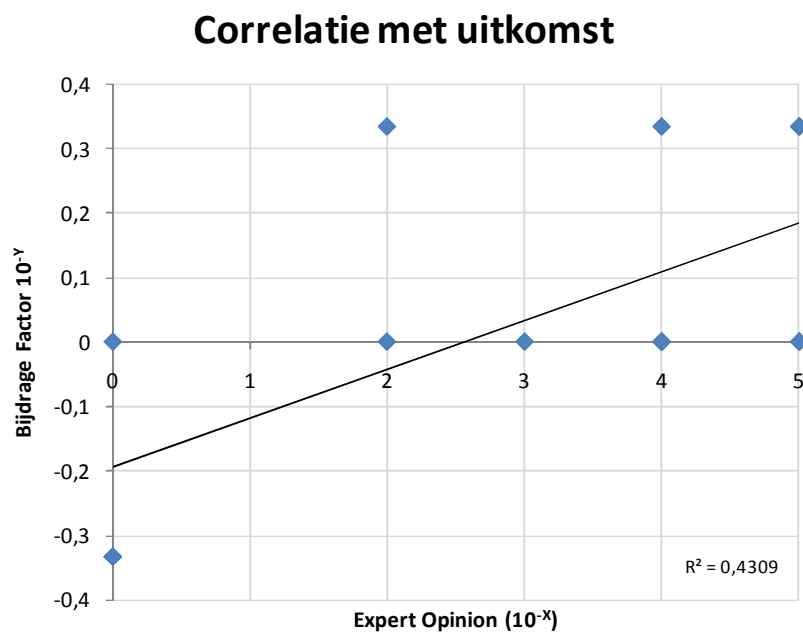
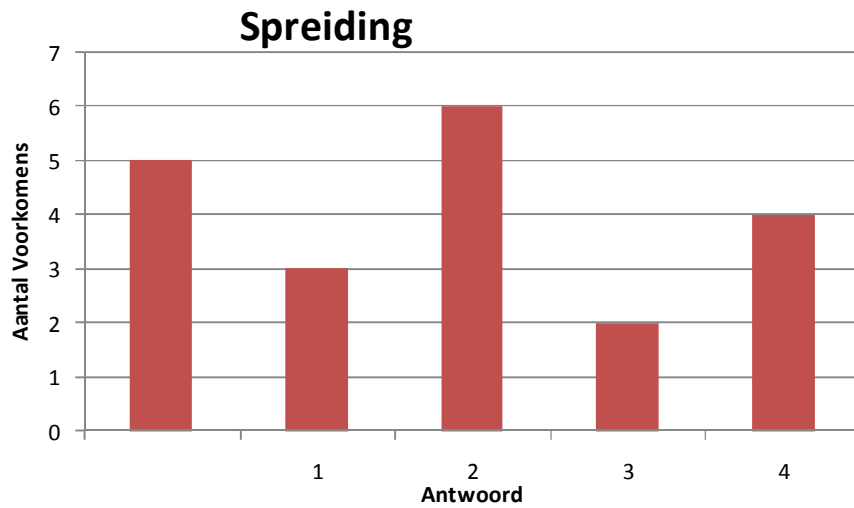
N=18

*Sterkte Architectuurconcept*



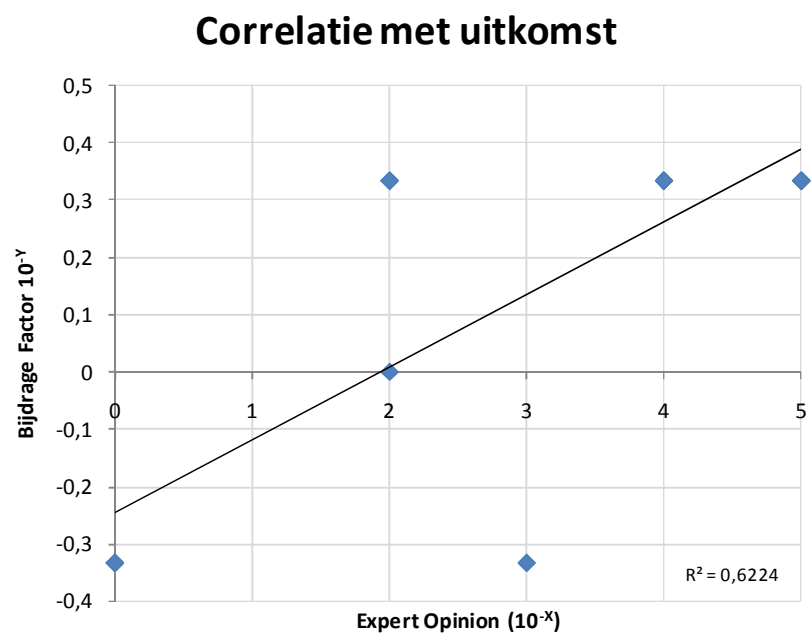
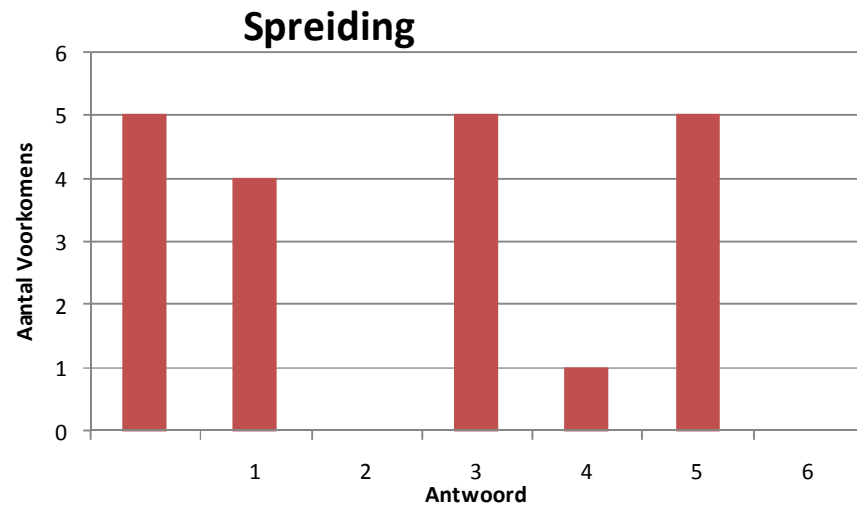
N=16

Gebruik van een vertrouwde/gecertificeerde compiler



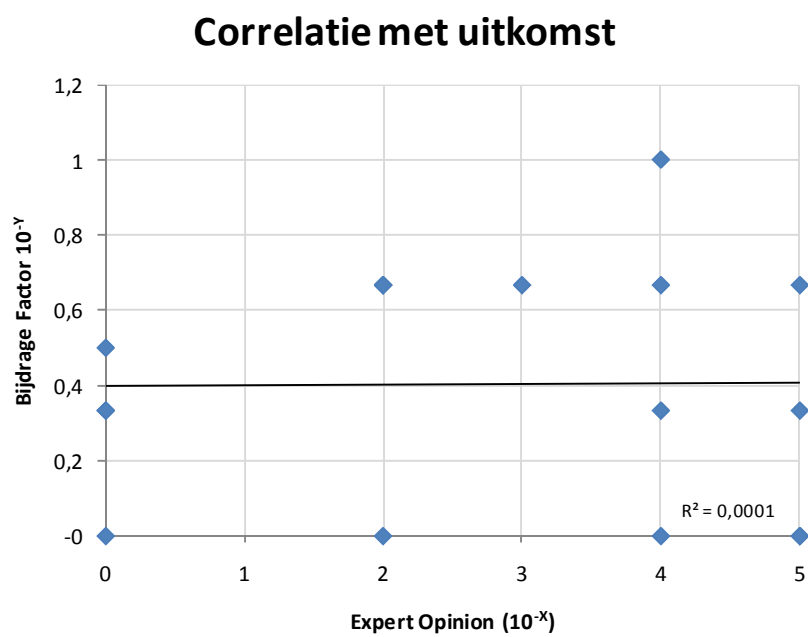
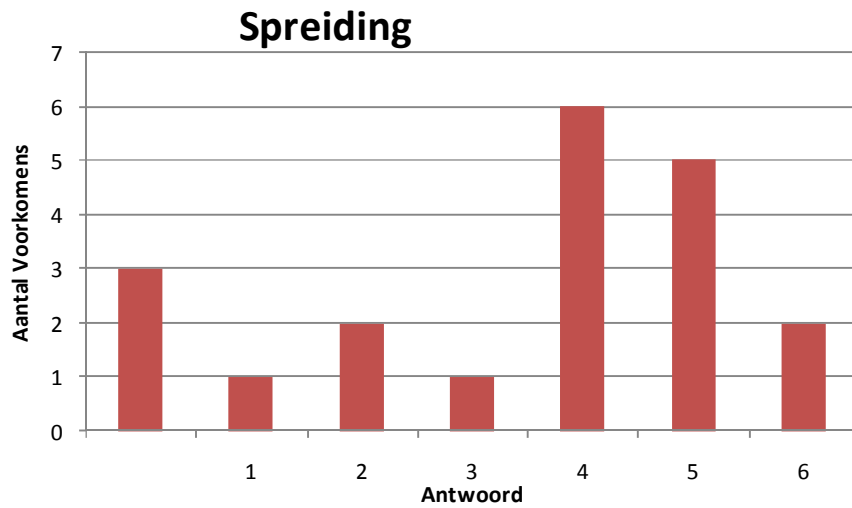
N=15

### Traceerbaarheid van requirements



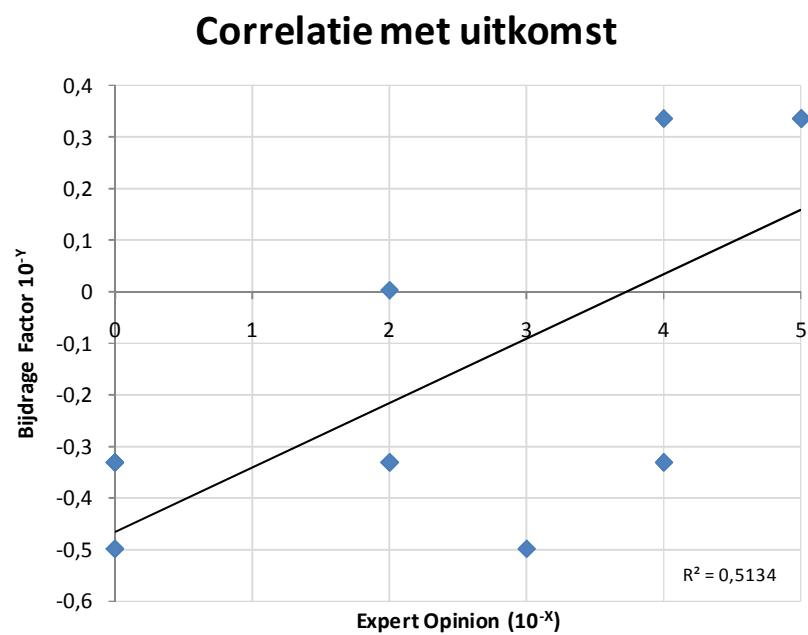
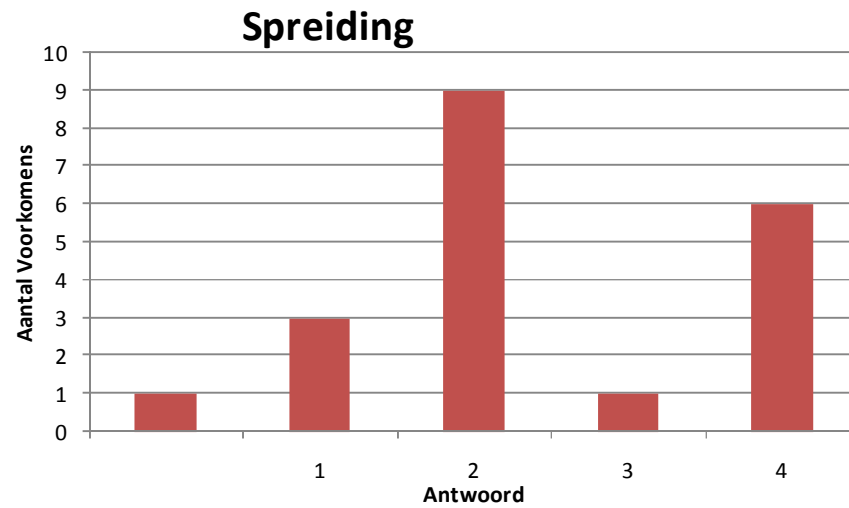
N=15

### Dekkingsgraad testen



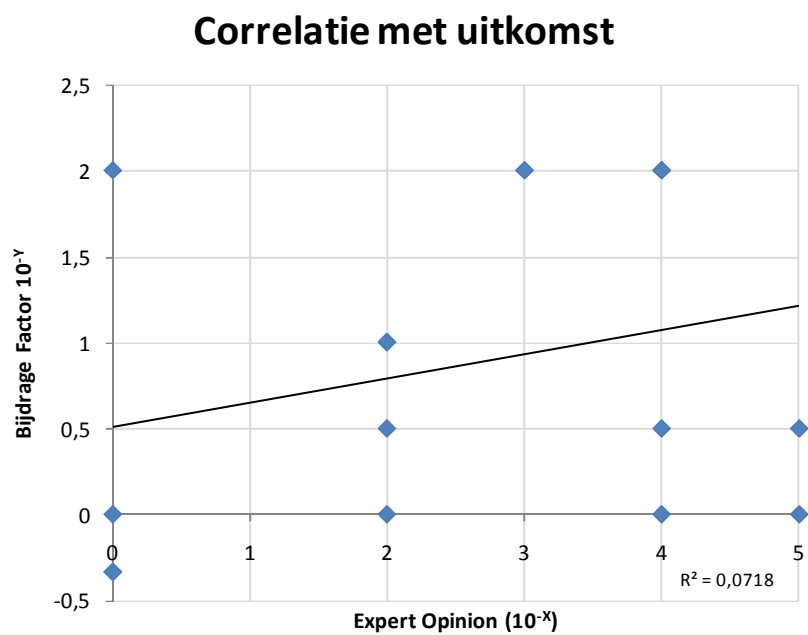
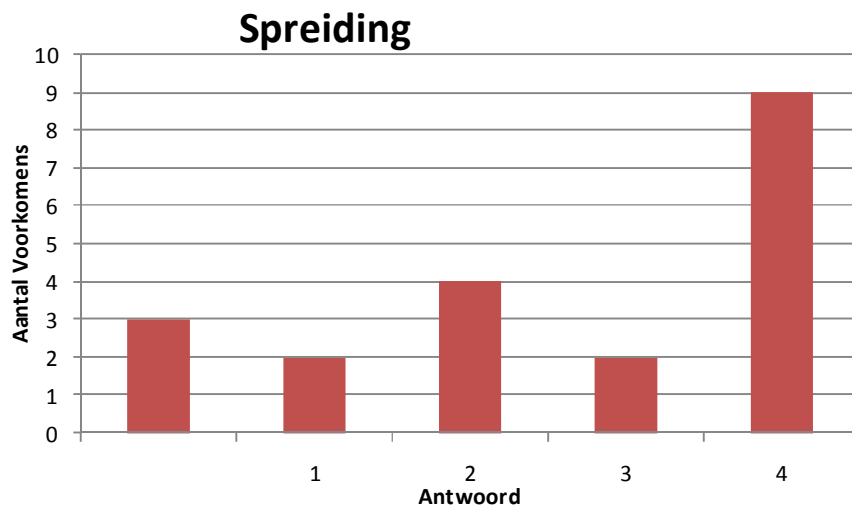
N=17

Run-time omgeving van de functie



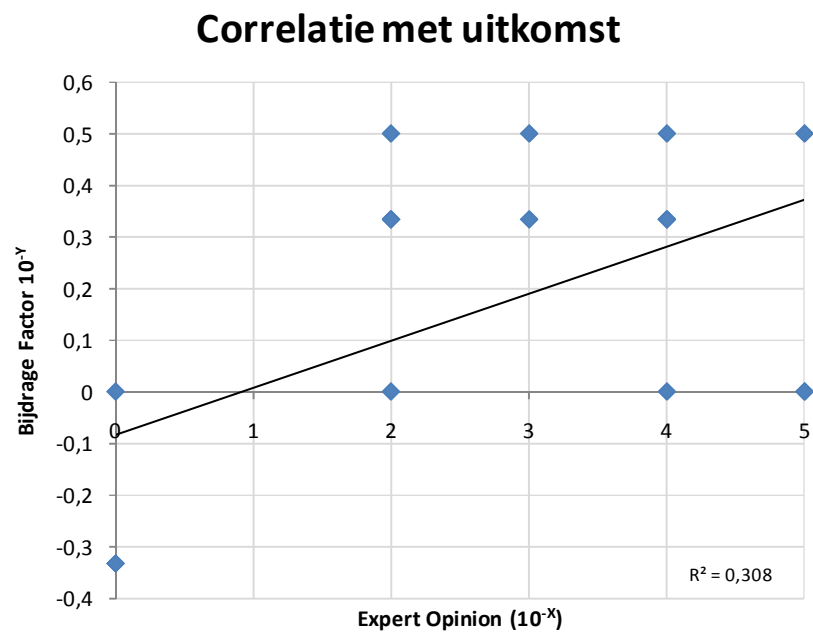
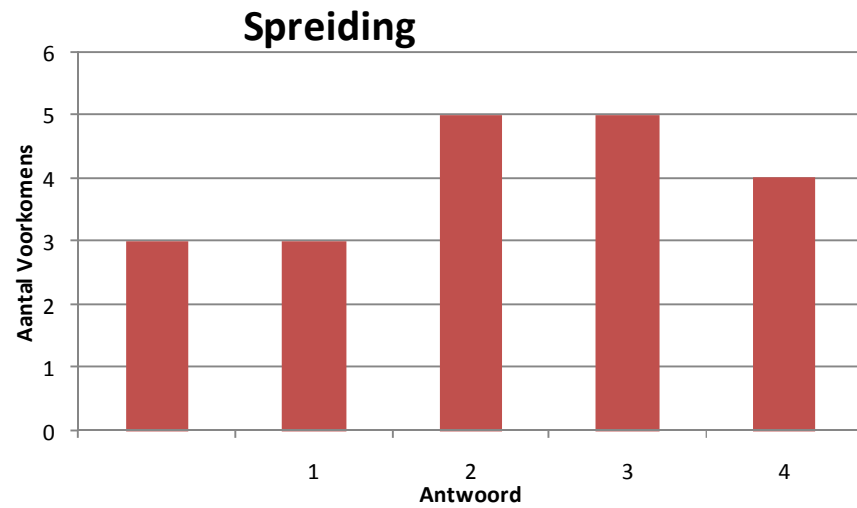
N=19

Aanwezigheid representatieve veldgegevens



N=17

## Monitoring



N=11

- [1] Sipke van Manen, Ed Brandt, Jaap van Ekris and Wouter Geurts, "*TOPAAS: An Alternative Approach to Software Reliability Quantification*", Quality and Reliability Engineering International, October 2013
- [2] Finetti, B. de, *Theory of Probability*, translation by A Machi and AFM Smith, 2 volumes, New York, Wiley, 1974-5
- [3] Savage, Leonard J., *The Foundations of Statistics*, Wiley, New York, 1954
- [4] Benjamin, J.R. and Cornell C.A., *Probability, Statistics and Decision for Civil Engineers*, McGraw-Hill, New York, 1970
- [5] Cooke, R.M., *Experts in Uncertainty*, Oxford University Press, New York, 1991
- [6] Scheff, Thomas et al., "*Goffman Unbound!: A New Paradigm for Social Science (The Sociological Imagination)*", Paradigm Publishers, 2006
- [7] Jang H., Ryu H., Kim J., Lee J. and Cho K., *A Probabilistic Risk Assessment for Field Radiography Based on Expert Judgment and Opinion*, *Progress in Nuclear Science and Technology*, Vol. 1, p.471-474, 2011
- [8] The European Food Authority. Opinion of the Scientific Panel on Animal Health and Welfare on a self mandate on the Framework for EFSA AHAW Risk Assessments, *The EFSA Journal* (2007), 550, 1-46
- [9] Almering, Vincent, Genuchten, Michiel van, Cloudt Ger and Sonnemans, Peter J.M., *Using Software Reliability Growth Models in Practice*, IEEE Software, ISSN: 0740-7459, Vol. 24, no. 6, 2007
- [10] M. Li, Y. Wei, D. Desovski, H. Nejad, S. Ghose, B. Cukic and C. Smidts, *Validation of a Methodology for Assessing Software Reliability*, *Proceedings of the 15th International Symposium on Software Reliability Engineering*, 2004
- [11] C. Smidts, M. Li, R.W. Brill, *Ranking Software Engineering Measures Related to Reliability Using Expert Opinion*, *Proceedings of the 11th International Symposium on Software Reliability Engineering*, 2000
- [12] C. Smidts and M. Li, *A Ranking of Software Engineering Measures Based on Expert Opinion*, *IEEE Transactions on Software Engineering*, Vol. 29, No. 9, September 2003
- [13] *Handreiking Prestatiegestuurde Risicoanalyses*, Rijkswaterstaat, versie 1.0.0, September 2016
- [14] *Handleiding kwantitatieve analyse menselijk handelen bij waterkeringen*, versie 1.0, Juli 2013.

# TOPAAS

|                             |                                                     |
|-----------------------------|-----------------------------------------------------|
| Nummer:                     | 1319                                                |
| Versienummer standaard:     | 2.0                                                 |
| Versienummer document:      |                                                     |
| Status:                     | In beheer                                           |
| Type:                       | Kader                                               |
| Inhoudelijk beheerder:      | Jeroen Horstman                                     |
| Verantwoordelijke afdeling: | Afd. IB Systeemtechniek                             |
| Netwerken:                  | Hoofdvaarwegennet, Hoofdwatersysteem, Hoofdwegennet |
| Rollen:                     | Technisch Manager                                   |
| Fase:                       | Realisatie                                          |
| Proceseigenaar:             | Proceseigenaar Aanleg en Onderhoud                  |
| Link om te reageren:        | <a href="#">Link</a>                                |